วิธีการจัดรูปแบบข้อมูลแบบใหม่สำหรับแก้ปัญหาระบบสมการเชิงเส้นโดยใช้วิธีเคราซ์ดีคอมโพสิชันบนระบบคลัสเตอร์

อนิรุธ สืบสิงห์

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาเทคโนโลยีสารสนเทศ คณะวิทยาศาสตร์
มหาวิทยาลัยอุบลราชธานี
พ.ศ. 2548

ISBN 974-523-027-8

A NEW DATA LAYOUT FOR SOLVING LINEAR EQUATIONS SYSTEM USING CROUT DECOMPOSITION ON A COMPUTATIONAL CLUSTER

ANIRUT SUEBSING

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF MASTER OF SCIENCE

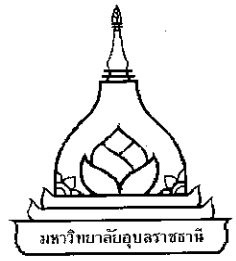
MAJOR IN INFORMATION TECHNOLOGY

FACULTY OF SCIENCES

UBON RAJATHANEE UNIVERSITY

YEAR 2005

ISBN 974-523-027-8



ใบรับรองวิทยานิพนธ์

มหาวิทยาลัยอุบลราชธานี

ปริญญา วิทยาศาสตรมหาบัณฑิต

สาขาวิชาเทคโนโลยีสารสนเทศ คณะวิทยาศาสตร์

เรื่อง วิธีการจัดรูปแบบข้อมูลแบบใหม่สำหรับแก้ปัญหาระบบสมการเชิงเส้นโดยใช้วิธีเคราซ์ดีคอมโพสิชันบนระบบคลัสเตอร์

ผู้วิจัย นายอนิรุธ สืบสิงห์

ได้พิจารณาเห็นชอบโดยคณะกรรมการสอบวิทยานิพนธ์

..... ประธานกรรมการ

(ผู้ช่วยศาสตราจารย์ ดร.อุทิศ อินทร์ประสิทธิ์)

..... กรรมการ

(ผู้ช่วยศาสตราจารย์ ดร.วนิดา แก่นอากาศ)

..... กรรมการ

(ดร.จิรดา กันทรารักษ์)

..... คณบดี

(ผู้ช่วยศาสตราจารย์ วรรณวไล อธิวาสน์พงศ์)

มหาวิทยาลัยอุบลราชธานี รับรองแล้ว

.....

(ศาสตราจารย์ ดร.ประกอบ วิโรจนกูฏ)

อธิการบดี มหาวิทยาลัยอุบลราชธานี

ปีการศึกษา 2548

กิตติกรรมประกาศ

วิทยานิพนธ์เล่มนี้สำเร็จได้เพราะได้รับความกรุณาจาก ดร.จิรดา กันทรารักษ์ ซึ่งเป็นอาจารย์ที่ปรึกษาที่คอยให้ความช่วยเหลือ ให้คำแนะนำ และข้อคิดเห็นในการศึกษาเป็นอย่างดีตลอดเวลาที่ได้ทำการศึกษา

ขอขอบคุณ ผู้ช่วยศาสตราจารย์ ดร.อุทิศ อินทร์ประสิทธิ์ หัวหน้าภาควิชาคณิตศาสตร์ สถิติและคอมพิวเตอร์ คณะวิทยาศาสตร์ มหาวิทยาลัยอุบลราชธานี ที่ให้คำปรึกษาชี้แนะเกี่ยวกับทฤษฎีทางคณิตศาสตร์

ขอขอบคุณคณาอาจารย์ และเจ้าหน้าที่ในภาควิชาคณิตศาสตร์ สถิติและคอมพิวเตอร์ทุกท่านที่ให้คำปรึกษาชี้แนะ และให้ความช่วยเหลือตลอดระยะเวลาการศึกษาในมหาวิทยาลัยอุบลราชธานี

ขอขอบพระคุณบิดา มารดาและครอบครัว ที่ให้กำลังใจในการศึกษาและการทำงานตลอดมา จึงทำให้วิทยานิพนธ์นี้สำเร็จลงได้ด้วยดี

(นายอนิรุธ สืบสิงห์)
ผู้วิจัย

บทคัดย่อ

ชื่อเรื่อง : วิธีการจัดรูปแบบข้อมูลแบบใหม่สำหรับแก้ปัญหาระบบสมการเชิงเส้นโดยใช้วิธีเคราซ์ดีคอมโพสิชันบนระบบคลัสเตอร์

โดย : อนิรุธ สืบสิงห์

ชื่อปริญญา : ปริญญาวิทยาศาสตรมหาบัณฑิต

สาขาวิชา : เทคโนโลยีสารสนเทศ [ISBN 974-523-027-8]

ประธานกรรมการที่ปรึกษา : ดร.จิรดา กันทรารักษ์

ศัพท์สำคัญ : การจัดรูปแบบข้อมูล คลัสเตอร์ สมการเชิงเส้น อัตราเร็ว ประสิทธิภาพ โปรแกรมประยุกต์แบบขนาน

วิทยานิพนธ์นี้ได้เสนอแนวคิดใหม่เกี่ยวกับการจัดรูปแบบข้อมูลสำหรับการประมวลผลแบบขนานบนระบบคลัสเตอร์ โดยให้ชื่อว่า ไรท์แองเกิลบล็อก เพื่อนำไปประยุกต์ใช้กับวิธีการแก้ปัญหาสมการเชิงเส้นด้วยวิธีการเคราซ์ดีคอมโพสิชัน ซึ่งจะช่วยเพิ่มประสิทธิภาพการทำงานได้ดีกว่าการจัดรูปแบบข้อมูลแบบคอลัมน์บล็อก และการจัดรูปแบบข้อมูลแบบแถวบล็อก โดยได้ทำการเปรียบเทียบประสิทธิภาพการทำงานแบบขนานของการจัดรูปแบบข้อมูลที่ออกแบบใหม่ กับการจัดรูปแบบข้อมูลแบบคอลัมน์บล็อก และการจัดรูปแบบข้อมูลแบบแถวบล็อก นอกจากนี้ยังได้ทำการเปรียบเทียบเวลาที่ใช้ในการประมวลแบบขนานด้วยการจัดรูปแบบข้อมูลแบบใหม่ กับการประมวลแบบลำดับด้วย ผลการวิจัยแสดงให้เห็นว่าการจัดรูปแบบข้อมูลแบบขนานรูปแบบใหม่ มีประสิทธิภาพการทำงานที่ดีกว่าการจัดรูปแบบข้อมูลแบบคอลัมน์บล็อก และการจัดรูปแบบข้อมูลแบบแถวบล็อกซึ่งเป็นการจัดรูปแบบข้อมูลรูปแบบเดิม และใช้เวลาในการประมวลน้อยกว่าการประมวลผลแบบลำดับ

ABSTRACT

TITLE : A NEW DATA LAYOUT FOR SOLVING LINEAR EQUATIONS SYSTEM USING CROUT DECOMPOSITION ON A COMPUTATIONAL CLUSTER

BY : ANIRUT SUEBSING

DEGREE : MASTER OF SCIENCE

MAJOR : INFORMATION TECHNOLOGY [ISBN 974-523-027-8]

CHAIR : JIRADA KUNTRARUK, Ph.D.

KEYWORDS : DATA LAYOUT / CLUSTER / LINEAR EQUATIONS / SPEEDUP/ EFFICIENCY / PARALLEL APPLICATION

This thesis proposes a new data layout design named Right-angle block that is used to solve the system of linear equations using Crout Decomposition method on a computational cluster. The proposed data layout shows the better performance compared with the column block and the row block layouts when executing the application in a distributed-memory parallel environment. We also compared the performance of a parallel execution using the Right-angle block data layout with a sequential execution. The parallel execution of the proposed data layout outperformed the sequential execution.

สารบัญ

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

หน้า

สารบัญภาพ

หน้า

ภาพที่

สารบัญภาพ (ต่อ)

หน้า

ภาพที่

สารบัญภาพ (ต่อ)

หน้า

ภาพที่

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ปัจจุบันในต่างประเทศได้มีการนำเอาระบบคลัสเตอร์ (Cluster) เข้ามาประยุกต์ใช้กับงานด้านต่าง ๆ ไม่ว่าจะเป็นภาครัฐหรือเอกชน ด้วยราคาของระบบคลัสเตอร์ที่ถูกกว่าเครื่องซุปเปอร์คอมพิวเตอร์และความสามารถในการเพิ่มความเร็วของการประมวลผลข้อมูลให้มากขึ้นได้เรื่อย ๆ เนื่องจากระบบคลัสเตอร์เป็นสถาปัตยกรรมที่นำเอาเครื่องพีซีทำมาการเชื่อมต่อเป็นเครือข่ายจึงทำให้สามารถเพิ่มเครื่องคอมพิวเตอร์เข้าไปได้อีกในภายหลัง ซึ่งจะแตกต่างจากซุปเปอร์คอมพิวเตอร์ที่ไม่สามารถจะทำการเพิ่มความเร็วในการประมวลผลได้อีกในภายหลัง นอกเสียจากจะทำการซื้อเครื่องใหม่ อีกทั้งระบบคลัสเตอร์ยังสามารถที่จะนำเอาเครื่องคอมพิวเตอร์ที่ตกรุ่นแล้วแต่ยังสามารถใช้งานได้ดีมาเชื่อมโยงในระบบคลัสเตอร์ทำให้เกิดประโยชน์ได้อีกครั้ง จึงทำให้หลาย ๆ หน่วยงานทั้งภาครัฐและเอกชนในต่างประเทศได้นำเอาระบบนี้เข้ามาช่วยในการทำงานแทนการซื้อเครื่องซุปเปอร์คอมพิวเตอร์ ในส่วนของประเทศไทยก็ได้มีการวิจัยและเริ่มนำระบบนี้เข้ามาใช้แล้ว ซึ่งมหาวิทยาลัยเกษตรศาสตร์ เป็นผู้ที่ริเริ่มนำระบบคลัสเตอร์เข้ามาใช้งานเป็นแห่งแรกในประเทศไทย เป็นที่รู้จักกันดีในนามของ SMILE Cluster [1]

อย่างไรก็ตาม ถึงแม้ว่าระบบคลัสเตอร์อาจจะมีความสามารถในการทำงานเทียบเท่ากับระบบซุปเปอร์คอมพิวเตอร์ แต่ซอฟต์แวร์ที่จะนำมาใช้กับระบบคลัสเตอร์ก็จะต้องเป็นซอฟต์แวร์ที่สามารถรองรับการประมวลผลแบบขนาน (Parallel Computing) ได้ ซึ่งพื้นฐานที่สำคัญสำหรับการที่จะพัฒนาซอฟต์แวร์ให้สามารถทำงานบนระบบคลัสเตอร์ได้อย่างมีประสิทธิภาพนั้น จะต้องเข้าใจถึงการจัดรูปแบบข้อมูล (Scatter Data Layout) เนื่องจากว่าระบบคลัสเตอร์ คือการนำเอาเครื่องคอมพิวเตอร์แบบพีซีหลาย ๆ เครื่องมาทำการเชื่อมต่อกันบนระบบเครือข่ายความเร็วสูง ดังนั้นในการที่จะทำให้เครื่องคอมพิวเตอร์ที่เชื่อมต่อกันอยู่สามารถที่จะช่วยประมวลผลข้อมูลได้ก็จำเป็นจะต้องหาวิธีการจัดรูปแบบข้อมูลที่เหมาะสม เพื่อทำการแบ่งย่อยงานเป็นชิ้นเล็ก ๆ แล้วจึงทำการกระจายงานเหล่านั้นไปยังเครื่องคอมพิวเตอร์บนระบบคลัสเตอร์อย่างเหมาะสม และการจัดรูปแบบข้อมูลที่ดีจะต้องทำให้ข้อมูลมีความเป็นอิสระต่อกันมากที่สุด เพราะการจัดสรรข้อมูลให้มีความอิสระต่อกันจะทำให้เครื่องคอมพิวเตอร์บนระบบคลัสเตอร์สามารถประมวลผลข้อมูลไปพร้อม ๆ

กันได้ ช่วยให้ประมวลผลงานเสร็จได้เร็วขึ้น โดยเทคนิคพื้นฐานที่นิยมใช้กันก็ได้แก่ การจัดรูปแบบข้อมูลแบบ Column Block [2, 3, 4] และการจัดรูปแบบข้อมูลแบบ Row Block [2, 3, 4]

ในงานวิจัยชิ้นนี้จะได้นำเสนอการจัดรูปแบบข้อมูลรูปแบบใหม่ที่ไม่ซ้ำกับรูปแบบเดิมที่มีอยู่แล้วขึ้นมาอีกหนึ่งรูปแบบ ซึ่งจะนำไปใช้แก้ไขปัญหาในเรื่องความไม่เป็นอิสระต่อกันของข้อมูลสำหรับการนำไปประยุกต์ใช้กับวิธีการแก้ปัญหาสมการเชิงเส้นด้วยวิธี Crout Decomposition เมื่อทำการประมวลผลแบบขนานบนระบบคลัสเตอร์ เนื่องจากวิธี Crout Decomposition [5, 6, 7, 8, 9] คือวิธีการแยกเมตริกซ์จัตุรัส (Triangular Matrix) A ให้อยู่ในรูปผลคูณของเมตริกซ์สามเหลี่ยมล่าง (Lower Triangular Matrix) L กับ เมตริกซ์สามเหลี่ยมบน (Upper Triangular Matrix) U โดยจะทำให้เมตริกซ์ U มีสมาชิกในแนวเส้นทแยงมุมเป็น 1 เมื่อได้เมตริกซ์ที่อยู่ในรูปผลคูณระหว่างเมตริกซ์ L กับ U แล้วก็จะนำเมตริกซ์ L กับ U ไปทำแก้สมการเพื่อหาค่าของตัวแปรที่ยังไม่ทราบค่าด้วยวิธีการ Forward และ Back Substitution ต่อไป ซึ่งวิธี Crout Decomposition นี้จะให้ความสำคัญกับการแยกเมตริกซ์จัตุรัส ให้อยู่ในรูปผลคูณระหว่างเมตริกซ์ L กับ U แต่เนื่องจากวิธี Crout Decomposition เป็นวิธีที่ไม่เหมาะสมเมื่อมีการนำไปใช้เพื่อพัฒนาเป็นโปรแกรมประยุกต์แบบขนานบนระบบคลัสเตอร์ เพราะจะทำให้ข้อมูลมีความเป็นอิสระต่อกันน้อย ส่งผลในทางลบต่อประสิทธิภาพในการทำงานจึงไม่เป็นที่นิยมนำวิธี Crout Decomposition ไปประยุกต์ใช้เพื่อทำการประมวลผลแบบขนาน แต่ด้วยรูปแบบของการจัดรูปแบบข้อมูลแบบใหม่ที่จะได้ทำการออกแบบในงานวิจัยนี้จะช่วยเพิ่มประสิทธิภาพในการทำงานสำหรับการประมวลผลแบบขนานบนระบบคลัสเตอร์ให้กับวิธี Crout Decomposition เมื่อเปรียบเทียบกับการจัดรูปแบบข้อมูลแบบ Column Block และแบบ Row Block

1.2 วัตถุประสงค์ของการวิจัย

1.2.1 เพื่อสร้างการจัดรูปแบบข้อมูลแบบใหม่ขึ้นมาอีก 1 รูปแบบ ที่สามารถช่วยแก้ไขปัญหาในเรื่องความไม่เป็นอิสระต่อกันของข้อมูลให้กับการแก้ปัญหาสมการเชิงเส้นด้วยวิธี Crout Decomposition ทำให้สามารถประมวลผลแบบขนานบนระบบคลัสเตอร์ได้อย่างมีประสิทธิภาพ

1.2.2 เพื่อทำการวิเคราะห์ และเปรียบเทียบให้เห็นถึงประสิทธิภาพของการจัดรูปแบบข้อมูลแบบใหม่ที่ได้ทำการสร้างขึ้น

1.3 ขอบเขตของการวิจัย

ขอบเขตการศึกษาในงานวิจัยนี้มุ่งเน้นที่จะสร้างการจัดรูปแบบข้อมูลแบบใหม่ขึ้นมา 1 รูปแบบ ที่สามารถจะนำไปใช้เพื่อช่วยแก้ไขปัญหาในเรื่องความไม่เป็นอิสระต่อกันของข้อมูลให้กับการแก้ปัญหาสมการเชิงเส้นด้วยวิธี Crout Decomposition ซึ่งจะช่วยเพิ่มประสิทธิภาพในการประมวลผลให้ดียิ่งขึ้น เมื่อทำการประมวลผลแบบขนานบนระบบคลัสเตอร์ โดยในการวัดประสิทธิภาพการทำงานจะทำการพิจารณาเปรียบเทียบจาก

1. ความเร็วของเวลาที่ใช้ในการประมวลผลข้อมูล (Execution Time) ระหว่างวิธีการจัดรูปแบบข้อมูลแบบใหม่ที่ประมวลผลแบบขนานบนระบบคลัสเตอร์ กับการประมวลผลแบบลำดับบนคอมพิวเตอร์ 1 เครื่อง

2. ความเร็วเร่ง (Speedup) ระหว่างการประมวลผลแบบขนานบนระบบคลัสเตอร์ที่ใช้การจัดรูปแบบข้อมูลแบบใหม่ กับการจัดรูปแบบข้อมูลแบบ Column Block และแบบ Row Block

โดยในการเปรียบเทียบจะทำการพัฒนาโปรแกรมประยุกต์ที่ใช้สำหรับแก้ปัญหาสมการเชิงเส้นด้วยวิธี Crout Decomposition ขึ้นมา ซึ่งโปรแกรมประยุกต์ที่จะได้ทำการพัฒนาขึ้นมาเพื่อใช้ในงานวิจัยนี้จะประกอบไปด้วย

1. โปรแกรมประยุกต์ที่ประมวลผลแบบลำดับ

2. โปรแกรมประยุกต์ที่ประมวลผลแบบขนานบนระบบคลัสเตอร์และใช้การจัดรูปแบบข้อมูลแบบ Column Block

3. โปรแกรมประยุกต์ที่ประมวลผลแบบขนานบนระบบคลัสเตอร์และใช้การจัดรูปแบบข้อมูลแบบ Row Block

4. โปรแกรมประยุกต์ที่ประมวลผลแบบขนานบนระบบคลัสเตอร์และใช้การจัดรูปแบบข้อมูลแบบใหม่

โดยในการทดลองจะได้ทำการทดลองโปรแกรมประยุกต์ที่ทำการประมวลผลผลแบบลำดับบนเครื่องคอมพิวเตอร์ 1 เครื่อง ในขณะที่โปรแกรมประยุกต์ที่ทำการประมวลผลแบบขนานจะถูกประมวลผลบนระบบคลัสเตอร์ขนาด 4 โหนด

1.4 เครื่องมือที่ใช้ทำการวิจัย

1.4.1 ฮาร์ดแวร์(Hardware)

- ชุดคอมพิวเตอร์เพนเทียมโฟร์ 2.40 กิกะเฮิร์ต (Pentium 4 2.41 GHz.) 5 เครื่อง
- การ์ดเน็ทเวิร์ค (Network Card) แบบ RJ-45 จำนวน 6 ตัว ความเร็ว 10/100 Mbps
- เน็ทเวิร์คสวิทช์ (Network Switch) จำนวน 1 ตัว ความเร็ว 10/100 Mbps

1.4.2 ซอฟต์แวร์ (Software)

- ระบบปฏิบัติการลีนุกส์ที่ใช้เคอร์เนลตั้งแต่เวอร์ชัน 2.4.x ขึ้นไป
- คอมไพเลอร์ภาษาซี/ซีพลัสพลัสที่ทำงานบนระบบปฏิบัติการลีนุกส์
- OSCAR 3.0
- ไลบารี่ MPI

1.5 สถานที่และระยะเวลาทำการวิจัย

ห้องปฏิบัติการคลัสเตอร์ ภาควิชาคณิตศาสตร์ สถิติ และคอมพิวเตอร์ คณะวิทยาศาสตร์ มหาวิทยาลัยอุบลราชธานี ระยะเวลาในการทำงานวิจัย จะเริ่มต้นตั้งแต่วันที่ 2 พฤศจิกายน พ.ศ. 2547 จนถึงวันที่ 18 มีนาคม พ.ศ. 2548 ดังในตารางที่ 1.1

ตารางที่ 1.1 ผังเวลาการดำเนินงาน

ลำดับ	กิจกรรม	สัปดาห์ที่									
		1-2	3-4	5-6	7-8	9-10	11-12	13-14	15-16	17-18	19-20
1	ศึกษางานวิจัยที่เกี่ยวข้อง										
2	ศึกษาวิธีการแก้ปัญหาสมการเชิงเส้น โดยใช้วิธี Crout Decomposition										
3	ศึกษาการจัดรูปแบบข้อมูลแบบ Column Block และแบบ Row Block ออกแบบวิธีการจัดรูปแบบข้อมูลแบบใหม่										
4	ศึกษาการพัฒนาโปรแกรมแบบ Parallel										
5	สร้างเป็นโปรแกรมประยุกต์เพื่อนำไปใช้ในการทดลอง										
6	ทำการทดลองและวัดประสิทธิภาพการทำงาน										
7	บันทึกผลการทดลองและสรุป										

1.6 ขั้นตอนในการดำเนินการวิจัย

1.6.1 ศึกษางานวิจัยที่เกี่ยวข้อง

1.6.2 ศึกษารูปวิธีแบบการแก้สมการเชิงเส้น โดยวิธี Crout Decomposition

1.6.3 ศึกษาการจัดรูปแบบข้อมูล แบบ Column Block และแบบ Row Block

1.6.4 ออกแบบวิธีการจัดรูปแบบข้อมูลแบบใหม่

1.6.5 ศึกษาวิธีการพัฒนาโปรแกรมแบบ Parallel

1.6.6 พัฒนาโปรแกรมที่ได้ทำการออกแบบไว้แล้ว

1.6.7 ทำการทดลองและวัดประสิทธิภาพการทำงาน

1.6.8 บันทึกผลการทดลองและสรุป

1.6.9 จัดทำเอกสาร

1.7 ประโยชน์ที่คาดว่าจะได้รับ

1.7.1 ได้วิธีการจัดรูปแบบข้อมูลแบบใหม่อีก 1 รูปแบบ ซึ่งสามารถจะช่วยแก้ไขปัญหาในเรื่องความไม่เป็นอิสระต่อกันของข้อมูลเมื่อทำการแก้ปัญหาสมการเชิงเส้นด้วยวิธี Crout Decomposition ทำให้การประมวลผลแบบขนานบนระบบคลัสเตอร์ มีประสิทธิภาพเมื่อทำการเปรียบเทียบกับการประมวลผลแบบขนานด้วยการจัดรูปแบบข้อมูลแบบ Column Block และแบบ Row Block

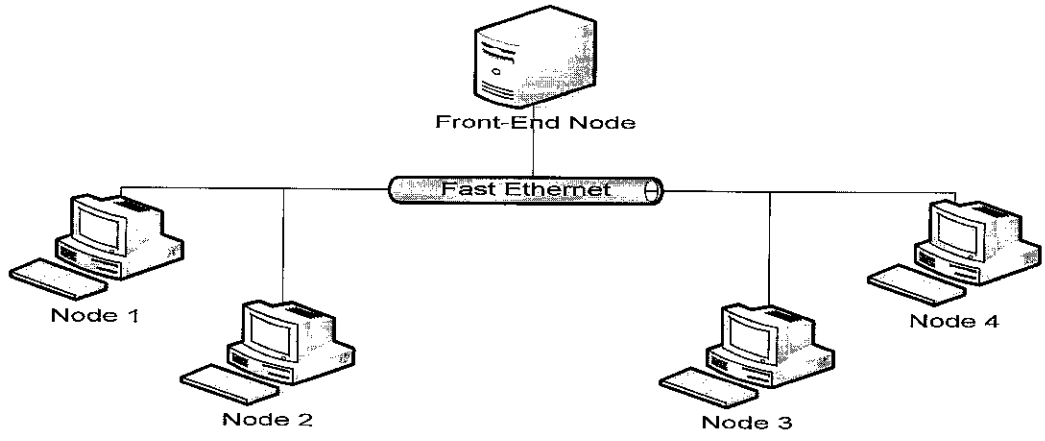
1.7.2 ได้โปรแกรมประยุกต์ที่ประมวลผลแบบขนานบนระบบคลัสเตอร์ ซึ่งใช้แก้ปัญหาสมการเชิงเส้นด้วยวิธี Crout Decomposition ที่มีประสิทธิภาพความเร็วในการประมวลผลมากกว่าโปรแกรมประยุกต์ที่ประมวลผลแบบลำดับ

บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 ความรู้พื้นฐานเกี่ยวกับระบบคลัสเตอร์

2.1.1 ระบบคลัสเตอร์คอมพิวเตอร์ (Cluster Computer System)

ภุชงค์ อุทโยภาศ [1, 10] ได้กล่าวเอาไว้ว่า ระบบคลัสเตอร์คอมพิวเตอร์ หมายถึงการนำคอมพิวเตอร์หลาย ๆ เครื่องมาเชื่อมต่อเข้าด้วยกันเพื่อให้ทำงานเหมือนเครื่องใหญ่เพียงเครื่องเดียว ความแตกต่างระหว่างระบบคลัสเตอร์ กับระบบ LAN ก็คือ ระบบ LAN จะเป็นการเชื่อมต่อคอมพิวเตอร์ เพื่อใช้ทรัพยากรร่วมกันโดยแต่ละเครื่องยังเป็นเครื่องอิสระ แต่ระบบคลัสเตอร์เป็นการรวมเครื่องพีซีหลายหลายเครื่องที่ประสานงานกันอย่างดีจนเปรียบเสมือนเป็นเครื่องเดียวกัน ระบบคลัสเตอร์นี้สามารถนำมาประยุกต์เพื่อใช้งานแทนระบบเซิร์ฟเวอร์ขนาดใหญ่ได้เป็นอย่างดีในราคาที่ถูกกว่า ดังภาพที่ 2.1



ภาพที่ 2.1 เป็นรูปโครงสร้างของระบบพีซีคลัสเตอร์ [1]

โครงสร้างของระบบพีซีคลัสเตอร์ทั่วไปจะประกอบด้วยเครื่องพีซีสมรรถนะสูงจำนวนหนึ่งที่ถูกนำมาเชื่อมต่อกันด้วยเครือข่ายความเร็วสูง เช่น ระบบ ATM Switch, Fast Ethernet Switch, Myrinet โดยโครงสร้างพื้นฐานของระบบคลัสเตอร์ และระบบ LAN นั้นไม่ได้มีความต่างกันมาก แต่ระบบคลัสเตอร์จะพึ่งพาซอฟต์แวร์พิเศษ ที่จะกระจายงานไปในกลุ่มของคอมพิวเตอร์ที่มารวมกันทำงานเป็นระบบคลัสเตอร์ ดังนั้นคุณสมบัติสำคัญของเทคโนโลยีคลัสเตอร์มีอยู่สามประการคือ เครือข่ายความเร็วสูง ระบบซอฟแวร์ที่สนับสนุนการทำงานแบบคลัสเตอร์ และโปรแกรมประยุกต์ที่ใช้ขีดความสามารถดังกล่าว

ปัจจุบันมีแนวโน้มว่าเทคโนโลยีหลาย ๆ เทคโนโลยีที่เริ่มสนับสนุนการก้าวไปสู่การใช้ระบบคลัสเตอร์ เช่น โพรเซสเซอร์ของเครื่องพีซีในปัจจุบันมีสมรรถนะแทบจะไม่แตกต่างจากโพรเซสเซอร์ของเครื่องซูเปอร์คอมพิวเตอร์ ดังนั้นการเชื่อมระบบพีซีเข้าด้วยกันจึงมีสมรรถนะไม่แตกต่างจากระบบซูเปอร์คอมพิวเตอร์ที่ขายกันอยู่ อย่างไรก็ตามเนื่องจากเทคโนโลยีของพีซีมีการขยายจำนวนมหาศาล ผลที่ได้คือราคาที่ถูกมากเมื่อเทียบกับประสิทธิภาพนอกจากนั้นการแข่งขันที่รุนแรงยังทำให้เงินทุนที่ทุ่มให้การวิจัยและพัฒนาเทคโนโลยีพีซีนั้นมหาศาล การพัฒนาจึงเป็นไปอย่างรวดเร็วทำให้ได้ของที่มีประสิทธิภาพสูงขึ้นมากในราคาเท่าเดิมปัญหาที่พบอยู่เสมอเมื่อใช้ระบบเซิร์ฟเวอร์ราคาแพงคือ ค่าบำรุงรักษาที่สูงมากเมื่อเทคโนโลยีนั้นเก่าหรือทำงานช้าไปแล้วนั้น การที่จะเพิ่มระบบจะเป็นไปได้ยากในขณะที่ในระบบพีซีคลัสเตอร์สามารถเพิ่มความสามารถได้ทีละน้อยในราคาถูกกว่า ข้อดีประการสำคัญก็คือระบบพีซีเป็นเทคโนโลยีที่คนส่วนใหญ่คุ้นเคยทำให้สามารถบำรุงรักษาระบบได้โดยไม่จำเป็นต้องจ้างบริษัทในราคาแพง

แนวความคิดของการนำระบบพีซีจำนวนมากมาต่อกันเป็นระบบใหญ่นั้นมีมานานมากแล้วแต่ความสนใจในการใช้งานเทคโนโลยีนี้เริ่มมาจากการกระตุ้นของโครงการแบวูฟล์ (Beowulf cluster) ขององค์การนาซ่า (NASA) ทำให้ระบบลีนุกส์คลัสเตอร์ (Linux Cluster) ถูกเรียกในอีกชื่อหนึ่งว่า แบวูฟล์คลัสเตอร์ด้วย Beowulf-style cluster นี้เป็นโครงการที่เกิดขึ้นมาในปี 1994 โดยนักวิจัยสองท่านคือ ดร.โทมัส สเตอร์ลิ่ง (Thomas Sterling) และโดนัล เบคเกอร์ (Don Becker) ซึ่งในขณะนั้นทำงานอยู่ที่ CESDIS (Center of Excellence in Space Data and Information Science) ซึ่งเป็นศูนย์วิจัยอยู่ที่ Goddard Space Flight Center ในรัฐแมรี่แลนด์ ประเทศสหรัฐอเมริกา โครงการนี้ได้รับสนับสนุนจากโครงการ ESS (Earth and Space Science Project) โดยมีจุดมุ่งหมายที่จะพัฒนาเทคนิคในการนำคอมพิวเตอร์แบบขนานขนาดใหญ่มาใช้แก้ปัญหาหรือจัดการกับข้อมูลขนาดใหญ่ที่ได้รับมาจากการสำรวจอวกาศแบวูฟล์คลัสเตอร์เครื่องแรกนั้นสร้างขึ้นมาจากเครื่อง PCs 486DX100 16 เครื่องที่เชื่อมต่อกันผ่านอีเธอเนต (channel-bonded 10 Mbps Ethernet) โครงการนี้เริ่มเป็นที่รู้จักมากขึ้นเมื่อมีการสาธิตในงาน Supercomputing 97 ซึ่งได้มีการเอานำเครื่อง PC Pentium Pro 200

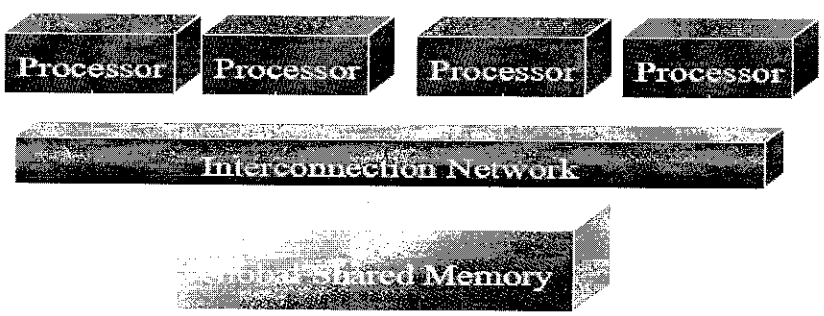
จำนวน 16 เครื่องมาต่อเข้าด้วยกันและพิสูจน์พลังให้เห็นโดยการคำนวณได้กว่า 1 พันล้านครั้งต่อวินาที ซึ่งเป็นสมรรถนะระดับซูเปอร์คอมพิวเตอร์อย่างแท้จริงในขณะนี้องค์การนาซ่าได้สร้างระบบคลัสเตอร์ขนาดใหญ่ไว้ใช้งานหลายระบบ เช่น Chiba city ซึ่งมีเครื่องพีซี 256 เครื่องมาต่อกัน แต่ระบบที่มีประสิทธิภาพสูงสุดได้แก่ CPlant ซึ่งเกิดจากการนำเครื่อง Alpha มาติดตั้งลีนุกส์แล้วต่อกันทั้งหมด 592 หน่วยประมวลผล ทำให้คำนวณได้กว่า 54.24 จิกะฟลอบ หรือกว่า 5 หมื่นล้านคำสั่งต่อวินาที และติดอันดับที่ 62 ของคอมพิวเตอร์ที่เร็วที่สุดในโลก

แบวูฟล์คลัสเตอร์เป็นระบบที่มีสถาปัตยกรรมประกอบด้วยเครื่องพีซีคอมพิวเตอร์หลาย ๆ ตัวซึ่งเชื่อมต่อเข้าด้วยกันโดยใช้เครือข่ายความเร็วสูง สถาปัตยกรรมแบวูฟล์คลัสเตอร์ต้องประกอบด้วยคุณสมบัติต่าง ๆ ดังนี้ อันดับแรกอุปกรณ์ที่ใช้ต่อเชื่อมในระบบแบวูฟล์คลัสเตอร์นี้ต้องเป็นของที่หาซื้อได้ง่ายตามท้องตลาดทั่วไป ประการที่สองซอฟต์แวร์ทั้งหมดที่จะนำมาใช้ต้องเป็นซอฟต์แวร์แบบ Open Source เช่น การเลือกใช้ระบบปฏิบัติการลีนุกส์ ประการถัดมาคือคอมพิวเตอร์แต่ละเครื่องต้องเชื่อมต่อกันด้วยเครือข่ายความเร็วสูงซึ่งเป็นประการสำคัญที่ทำให้ระบบมีความแตกต่างจากระบบ LAN ทั่วไป และประการสุดท้ายก็คือภายในระบบต้องมี Software สำหรับใช้ในการประมวลผลแบบขนานหรือกระจายที่ทำให้สามารถดึงกำลังเครื่องมาใช้งานได้

2.1.2 โครงสร้างคอมพิวเตอร์แบบขนาน

ในบทความของ ภุชงค์ อุทโยภาศ [1, 10] จำแนกโครงสร้างคอมพิวเตอร์แบบขนานออกเป็น 2 แบบใหญ่ ๆ ได้แก่

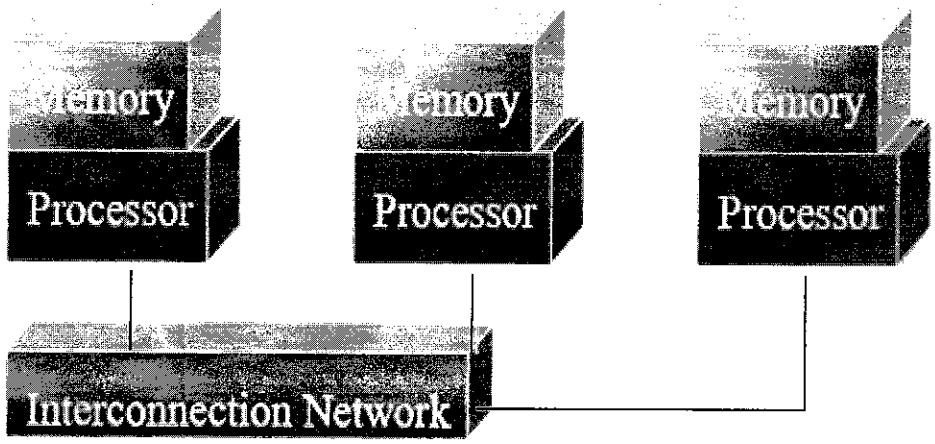
2.1.2.1 โครงสร้างแบบที่ใช้หน่วยความจำร่วม (Shared Memory Multiprocessors) เช่น เครื่องคอมพิวเตอร์ที่เป็นแบบ Dual processor หรือเครื่อง Multiprocessor โครงสร้างของเครื่องแบบใช้หน่วยความจำร่วมจะแสดงได้ ดังภาพที่ 2.2



ภาพที่ 2.2 โครงสร้างของเครื่องแบบใช้หน่วยความจำร่วม [1]

เครื่องคอมพิวเตอร์แบบนี้ประกอบไปด้วยส่วนประมวลผลจำนวนมากที่เชื่อมต่อกันด้วยเครือข่ายภายในความเร็วสูง (High Speed Interconnection network) โดยที่หน่วยประมวลผลทั้งหมดใช้หน่วยความจำกลางร่วมกัน ซึ่งข้อดีของเครื่องแบบนี้ก็คือการโปรแกรมง่าย และเครื่องมีความเร็วสูงเมื่อมีข้อมูลมีขนาดเล็กถึงกลางแต่ขีดจำกัดที่สำคัญคือโครงสร้างแบบนี้ไม่สามารถขยายตัวเป็นระบบขนาดใหญ่ได้เนื่องจากปริมาณการสื่อสารระหว่างโปรเซสเซอร์มีจำนวนมหาศาลทำให้เครือข่ายถึงจุดอิ่มตัวอย่างรวดเร็วเมื่อเพิ่มหน่วยประมวลผล

2.1.2.2 โครงสร้างแบบที่ใช้หน่วยความจำแยก (Distributed Memory Multicomputer) ซึ่งของคอมพิวเตอร์ที่ใช้โครงสร้างแบบนี้ได้แก่ เครื่องซุปเปอร์คอมพิวเตอร์ของ IBM รุ่น IBMSP คอมพิวเตอร์แบบนี้ประกอบไปด้วยส่วนประมวลผลจำนวนมากที่เชื่อมต่อกันด้วยเครือข่ายความเร็วสูง (High Speed Interconnection Network) โดยที่หน่วยประมวลผลแต่ละตัวจะมีหน่วยความจำแยกจากกันโดยเด็ดขาดการสื่อสารทำโดยการแลกเปลี่ยน message ผ่านเครือข่ายบางครั้งเครื่องคอมพิวเตอร์แบบนี้จึงถูกเรียกว่าเครื่องคอมพิวเตอร์แบบส่งผ่านข้อความ (Message Passing Parallel Computer) ข้อดีของคอมพิวเตอร์แบบนี้คือปริมาณการสื่อสารผ่านเครือข่ายจะน้อยกว่าแบบใช้หน่วยความจำร่วมมากทำให้เครื่องสามารถขยายตัวจนมีขนาดใหญ่มากได้ (scalable) โครงสร้างของเครื่องแบบใช้หน่วยความจำแยกจะแสดงได้ ดังภาพที่ 2.3

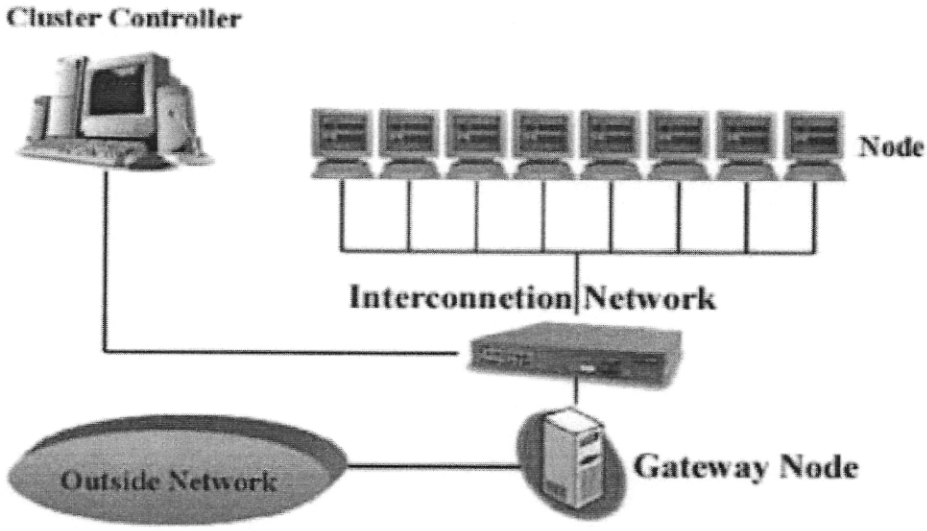


ภาพที่ 2.3 โครงสร้างของเครื่องแบบใช้หน่วยความจำแยก [1]

2.1.3 รูปแบบของระบบคลัสเตอร์

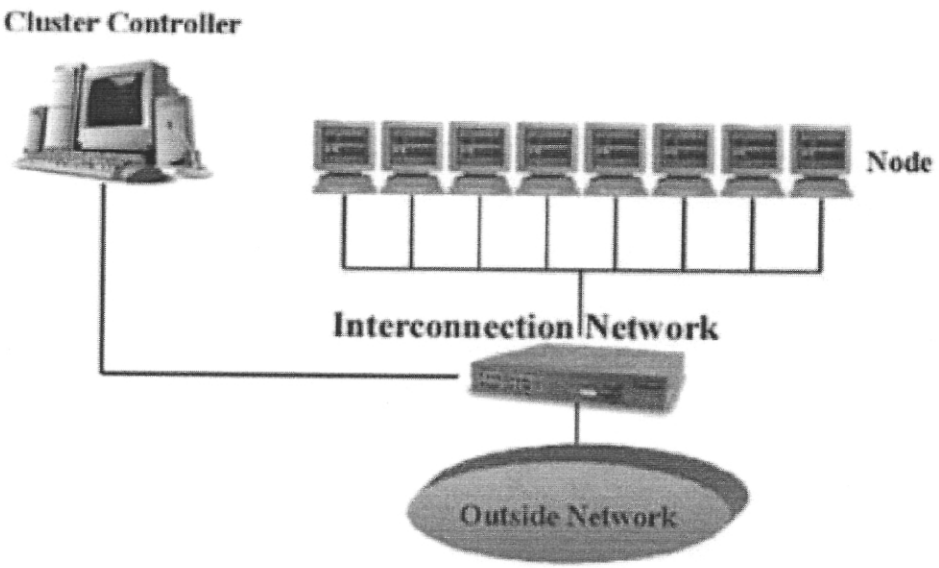
ในบทความของ ภุชงค์ อุทโยภาศ [1, 10] ได้แบ่งรูปแบบของระบบคลัสเตอร์ออกเป็น 2 รูปแบบใหญ่ คือ

2.1.3.1 ระบบคลัสเตอร์แบบเปิด ซึ่งระบบนี้จะต่อระบบคลัสเตอร์ผ่านเกตเวย์ที่ซ่อนทั้งระบบเครือข่ายภายนอก ข้อดีก็คือมีความปลอดภัยสูง ข้อเสียก็คือแต่ละโหนดในระบบคลัสเตอร์จะไม่สามารถช่วยกันบริการข้อมูลกับภายนอกได้ จึงเหมาะกับการใช้งานคำนวณขนาดใหญ่ ๆ เท่านั้น โครงสร้างของระบบคลัสเตอร์แบบปิดจะเป็น ดังภาพที่ 2.4



ภาพที่ 2.4 โครงสร้างของระบบคลัสเตอร์แบบปิด [1]

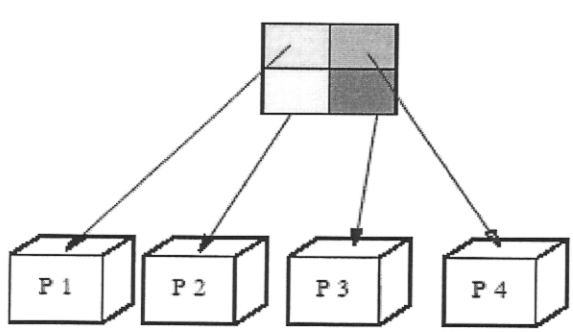
2.1.3.2 ระบบคลัสเตอร์แบบปิด ระบบนี้จะต่อกับเน็ตเวิร์คภายนอกโดยตรงทำให้ผู้ใช้เข้าถึงทุกโหนดในระบบคลัสเตอร์ได้โดยตรง ข้อเสียคือความปลอดภัยจะต่ำลงมากเพราะต้องคอยดูแลทุกเครื่องในระบบ แต่ข้อดีก็คือระบบสามารถช่วยกันบริการข้อมูลได้ จึงเหมาะกับงานบริการข่าวสารจำนวนมากเช่นทำหน้าที่เป็นระบบเซิร์ฟเวอร์โครงสร้างของระบบคลัสเตอร์แบบเปิด ดังภาพที่ 2.5



ภาพที่ 2.5 โครงสร้างของระบบคลัสเตอร์แบบเปิด [1]

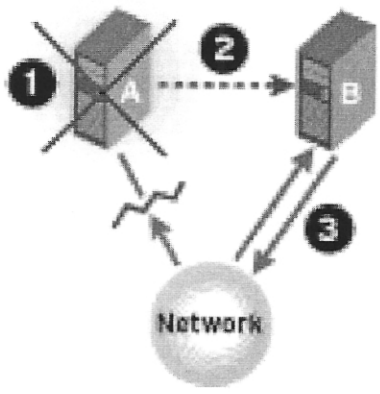
2.1.4 จุดประสงค์หลักของการนำเทคโนโลยีระบบคลัสเตอร์มาใช้งาน

2.1.4.1 เพื่อเร่งความเร็วในการงานหรือทำให้ระบบรองรับผู้ใช้ได้มากขึ้นในราคาที่ต่ำลง ซึ่งจะใช้เทคโนโลยีของการประมวลผลแบบขนานหรือแบบกระจายโดยจะทำการแบ่งงานใหญ่ ๆ ออกเป็นงานเล็ก ๆ จำนวนมากจากนั้นจะให้เครื่องหลายเครื่องช่วยกันประมวลผล เช่น การทำระบบเว็บเซิร์ฟเวอร์ที่ขยายขนาดได้มากทำได้โดยการกระจายการเรียกค้นของผู้ใช้ไปหลาย ๆ เครื่องจะทำผู้ใช้รู้สึกได้ทันทีว่าระบบเร็วขึ้นเนื่องจากแต่ละเครื่องรับงานน้อยลง



ภาพที่ 2.6 แนวคิดของการประมวลผลแบบขนาน [1, 10]

2.1.4.2 ทำให้ระบบมีความน่าเชื่อถือสูงขึ้นเพื่อรองรับงานที่เป็นแบบ Mission Critical Operation เช่น การให้บริการฐานข้อมูลที่ต้องการระบบที่ทำงานได้ตลอด 24 ชั่วโมงตลอดปี การประยุกต์หลักการของระบบคลัสเตอร์ อาศัยการทำงานประสานกันของระบบจำนวนมากเมื่อเครื่องใดเครื่องหนึ่งหยุดทำงาน งานต่าง ๆ จะถูกโอนย้าย หรือถูกทดแทนโดยอัตโนมัติด้วยเครื่องอื่น ๆ ทำให้ผู้ใช้รู้สึกว่าเครื่องทำงานอยู่ตลอดเวลา



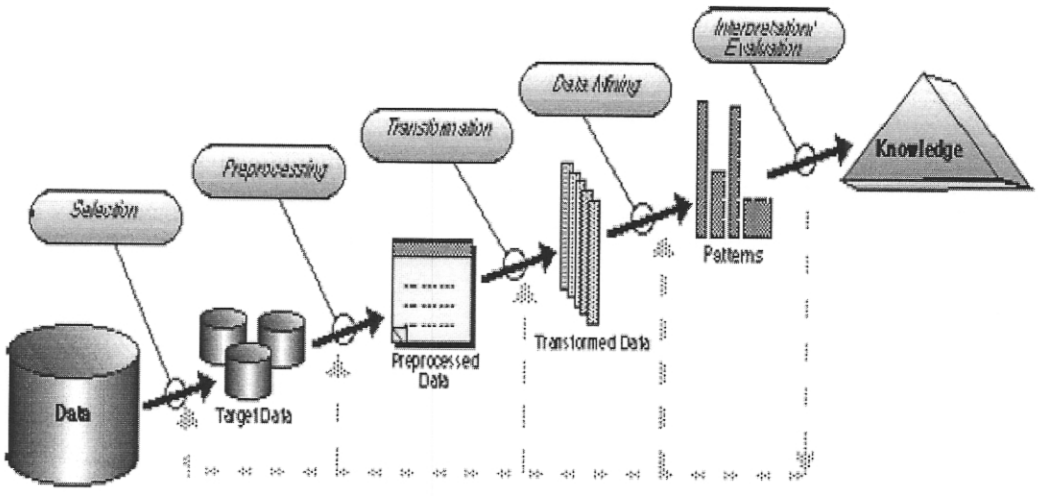
ภาพที่ 2.7 แสดงถึงระบบที่มีความหน้าเชื่อถือสูง (High Availability) [1, 10]

2.1.5 การประยุกต์ใช้งานลีนุกส์คลัสเตอร์

ระบบลีนุกส์คลัสเตอร์สามารถนำไปประยุกต์ได้หลายด้านทั้งทางด้านธุรกิจและทางด้านการประมวลผลทางวิทยาศาสตร์ซึ่ง ภุชงค์ อุทโยภาศ [1, 10] ได้ยกตัวอย่างไว้ดังต่อไปนี้

2.1.5.1 การประยุกต์ใช้งานลีนุกส์คลัสเตอร์ในงานด้านธุรกิจเนื่องจากการประยุกต์ใช้เทคโนโลยีทางคอมพิวเตอร์กับธุรกิจเริ่มมีมากขึ้น ดังนั้นการใช้งานของลีนุกส์คลัสเตอร์จึงมิได้ถูกจำกัดอยู่เพียงงานทางด้านวิทยาศาสตร์เท่านั้นแต่สามารถนำไปประยุกต์ใช้กับการใช้งานทางด้านธุรกิจได้ด้วยซึ่งพอจะแบ่งเป็นด้านต่าง ๆ ได้ดังนี้

1) ด้านการประมวลผลข้อมูลขนาดใหญ่ ในการคำนวณข้อมูลที่มีขนาดใหญ่เพื่อให้ได้มาซึ่งฐานความรู้ (Knowledge Based) นั้นจำเป็นต้องอาศัยเทคนิคต่าง ๆ เช่น การทำ Information Retrieval, Data Mining, Data Warehouse, การทำ index ของเอกสารต่าง ๆ ซึ่งเป็นเทคนิคที่อยู่เบื้องหลังของความสำเร็จของเว็บเซิร์ฟเวอร์ในปัจจุบันซึ่งการกระทำดังกล่าวจำเป็นต้องอาศัยเครื่องคอมพิวเตอร์สมรรถนะสูงเนื่องจากเป็นงานที่ใช้เวลาในการทำงานหลายวัน

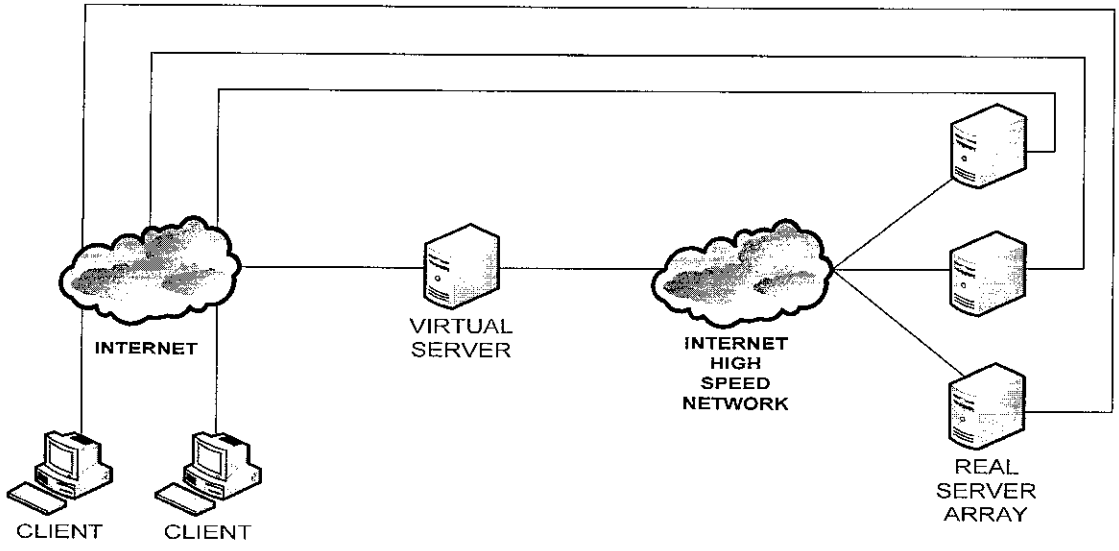


ภาพที่ 2.8 แสดงถึงกระบวนการของ Data Mining [10]

2) ด้านการสร้างเว็บเซิร์ฟเวอร์ จุดประสงค์ของการนำลีนุกส์มาใช้งานเป็นเว็บเซิร์ฟเวอร์ก็เพื่อทำให้เว็บนั้นมีคุณสมบัติในด้าน High-Availability ซึ่งหมายถึงการทำให้ User สามารถใช้บริการตลอดเวลาถึงแม้จะมีบางโหนด (Node) ได้หยุดทำงานไปก็ตาม ซึ่งการใช้งานลีนุกส์คลัสเตอร์เป็นเว็บเซิร์ฟเวอร์จะต้องใช้ระบบคลัสเตอร์แบบเปิดเพื่อให้ User ติดต่อเข้ามาผ่านทางอินเตอร์เน็ตได้การใช้งานลีนุกส์คลัสเตอร์เป็นเว็บคลัสเตอร์ที่สามารถขยายตัวได้ต้องอาศัยซอฟต์แวร์ช่วยในการกระจายงานโดยในเบื้องต้นอาจใช้วิธีการกระจายงานแบบง่าย ๆ โดยแก้ไข Domain name Server (DNS) ให้ชื่อเดียวกันแต่มีหลาย IP ทำให้เมื่อมีไคลแอนต์ต้องการ Resolve Hostname เป็น IP DNS จะตอบโดยกระจาย IP ไปบนหลายเครื่องทำให้การบริการผู้ใช้ถูกกระจายออกตามไปด้วยการติดตั้งอีกแบบที่ดีกว่าแต่ซับซ้อนกว่าคือการใช้ Linux Virtual Server (LVS) ซึ่งเป็นการแก้ไขพิเศษในระดับเคอร์เนล (Kernel) ซึ่งใน RedHat 6.2 สนับสนุน LVS เป็นที่เรียบร้อยแล้ว ซอฟต์แวร์ LVS จะทำงานโดยบนเซิร์ฟเวอร์โหนดดังนี้

1. รับ IP based Request จากไคลแอนต์
2. เลือกเซิร์ฟเวอร์ที่เหมาะสมเพื่อให้บริการไคลแอนต์นั้น
3. ส่ง IP โดยผ่านการแปลงแอดเดรสให้เหมาะสมไปยังเซิร์ฟเวอร์ที่ถูกเรียก
4. เซิร์ฟเวอร์ค้นและส่งข้อมูลกลับไปให้ไคลเอนต์เอง

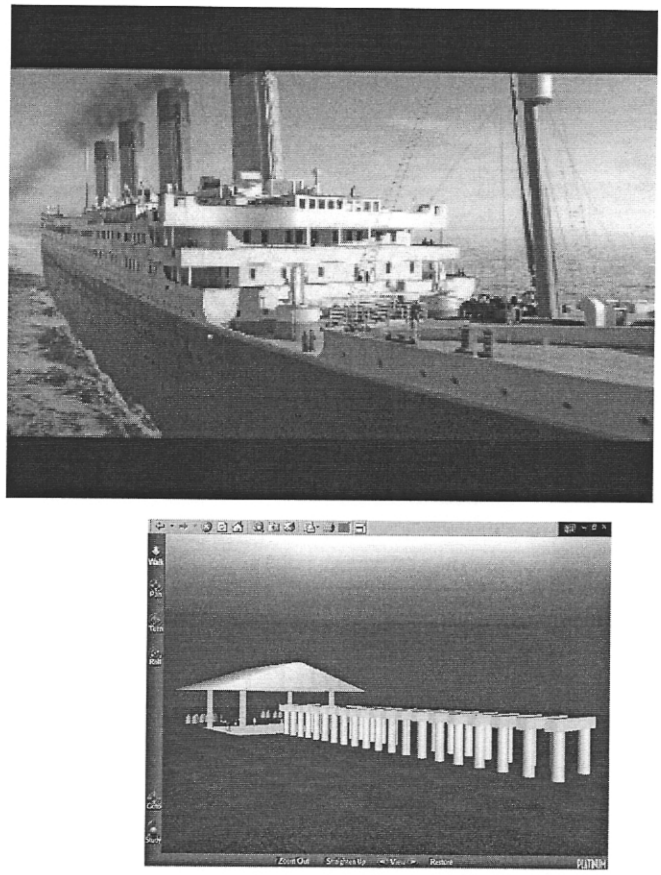
ในปัจจุบันมีเว็บเซิร์ฟเวอร์หลายตัวใช้วิธีนี้เช่น http://www.google.com ซึ่งใช้ลีนุกส์คลัสเตอร์มากกว่า 4,000 โหนด เพื่อใช้ในการให้บริการการค้นข้อมูลบนอินเตอร์เน็ต LVS



ภาพที่ 2.9 การทำ Web Clustering โดยใช้ LVS [10]

3) ด้านการสร้าง Internet Server ขนาดใหญ่ ภายในองค์กร ในอดีตการใช้งาน Internet Server ขนาดใหญ่นิยมใช้เซิร์ฟเวอร์เครื่องเดียวที่มีสมรรถนะสูงเพื่อรองรับจำนวนของผู้ใช้บริการที่เพิ่มขึ้นเรื่อย ๆ แต่วิธีการดังกล่าวต้องใช้ค่าใช้จ่ายสูง ดังนั้นลีนุกส์คลัสเตอร์ซึ่งมีความสามารถในการให้บริการด้าน Internet Server อยู่แล้วจึงเป็นทางออกที่ดีสำหรับผู้ให้บริการ Internet Server ขนาดใหญ่ซึ่งสามารถนำไปประยุกต์ใช้งานใน 2 ด้านคล้ายกับเว็บเซิร์ฟเวอร์ดังที่ได้กล่าวไปแล้วอีกด้านก็คือการใช้ทำหน้าที่เป็น File Server เพื่อที่จะ Share ข้อมูลระหว่างพนักงานภายในองค์กรเพื่อประหยัด Hard Disk ซึ่งจำเป็นต้องใช้เซิร์ฟเวอร์ที่มีขนาดใหญ่เพื่อที่จะตอบสนองผู้ใช้ได้อย่างรวดเร็ว ดังนั้นลีนุกส์เซิร์ฟเวอร์ก็เป็นทางออกที่ดีสำหรับการนำไปประยุกต์ใช้ในด้านนี้เช่นกัน

4) ด้านธุรกิจการสร้างภาพยนตร์ ในการสร้างภาพยนตร์ที่ใช้ Computer Graphics เพื่อสร้างความสมจริงให้กับภาพยนตร์นั้นจะต้องใช้เวลาในการสร้างภาพ (Render) ในแต่ละเฟรมค่อนข้างนานซึ่งกว่าจะได้ภาพยนตร์สักเรื่องหนึ่งต้อง Render ภาพหลายเฟรมมาก ดังนั้นจึงมีผู้ประยุกต์ใช้เทคโนโลยีลีนุกส์คลัสเตอร์เข้ากับการสร้างภาพยนตร์เช่น บริษัท Digital Domain ผู้สร้างความสมจริงให้กับภาพยนตร์ที่รู้จักหลายเรื่อง เช่น Titanic, True Lies, Apollo 13, Dante's Peak, Fifth Element เป็นต้นได้ใช้ลีนุกส์คลัสเตอร์ 160 เครื่องเพื่อการสร้างภาพที่สมจริงดังกล่าว



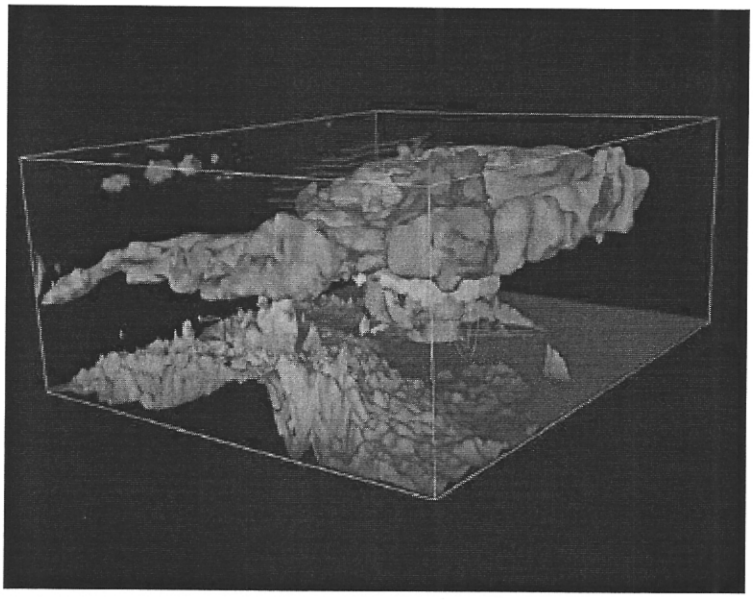
ภาพที่ 2.10 ตัวอย่างการ Render ภาพกราฟิกด้วยลีนุกส์คลัสเตอร์ [10]

5) ด้านการสร้างร้านค้าเสมือน ถ้าสามารถเลือกซื้อสินค้าโดยที่สามารถเดินเข้าไปในร้านค้าเสมือนที่อยู่ใน 3 มิติแล้วสามารถเลือกซื้อสินค้าดังกล่าวแบบ On-demand เช่น ถ้าเดินเข้าไปในเว็บเพจของร้านให้เช่าวีดีโอแล้วคลิกที่ภาพยนตร์แล้วมีภาพยนตร์ที่ส่งผ่านอินเตอร์เน็ตมาให้ดูถึงบ้าน ซึ่งการกระทำดังกล่าวจำเป็นต้องใช้คอมพิวเตอร์ความเร็วสูงที่ใช้ในการส่งข้อมูลมาให้ลูกค้า ดังนั้นลีนุกส์คลัสเตอร์ก็เป็นอีกหนึ่งทางเลือกที่น่าสนใจที่จะประยุกต์ใช้ในด้านดังกล่าว

2.1.5.2 การประยุกต์ใช้งานลีนุกส์คลัสเตอร์ในงานด้านวิทยาศาสตร์ ในปัจจุบันหลายหน่วยงานเริ่มนิยมใช้ลีนุกส์คลัสเตอร์ในการคำนวณงานทางด้านวิทยาศาสตร์เพื่อทดแทนระบบซุปเปอร์คอมพิวเตอร์งานดังกล่าวสามารถแบ่งเป็นกลุ่มได้ดังนี้

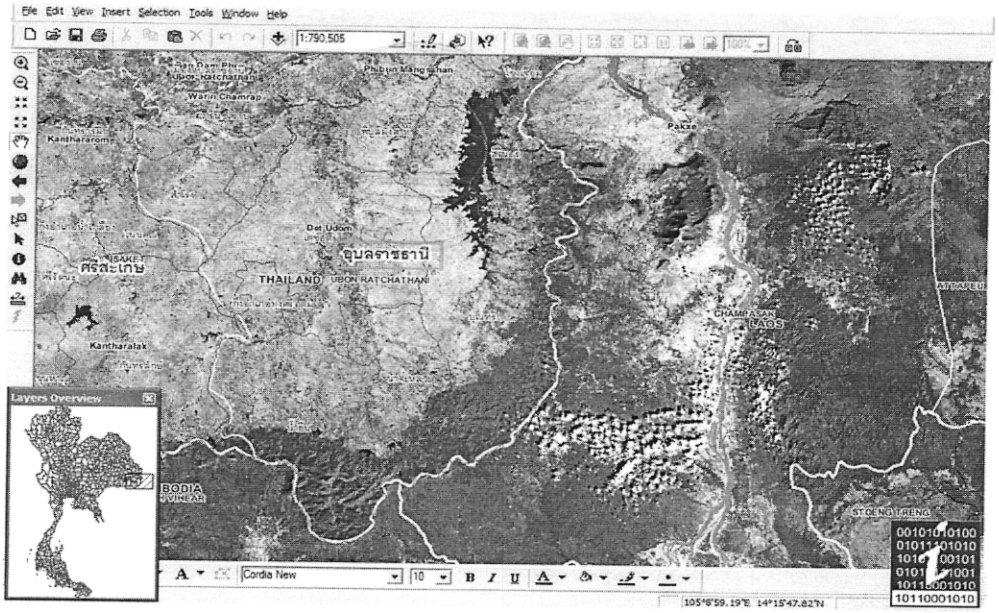
1) ด้านการพยากรณ์อากาศ ในอดีตในการพยากรณ์อากาศนั้นจำเป็นต้องใช้ระบบซุปเปอร์คอมพิวเตอร์เนื่องจากต้องอาศัยการคำนวณอย่างมหาศาลในเวลาอันจำกัดแต่อย่างไรก็ตามระบบซุปเปอร์นั้นมีราคาสูงมากดังนั้นการใช้ลีนุกส์คลัสเตอร์จึงเป็นทางออกที่ดีทางหนึ่งสำหรับประเทศยากจนในการพยากรณ์อากาศ ในปัจจุบันมีผู้พัฒนาโปรแกรมในการพยากรณ์อากาศ

สำหรับลีนุกส์คลัสเตอร์ เช่น โปรแกรม Parallel PSU/NCAR Mesoscale Model Generation 5 (MM5) ในปัจจุบันมีผู้นิยมใช้ซอฟต์แวร์ดังกล่าวมาใช้ในสถาบันทางด้านพยากรณ์อากาศมากกว่า 30 ประเทศทั่วโลก



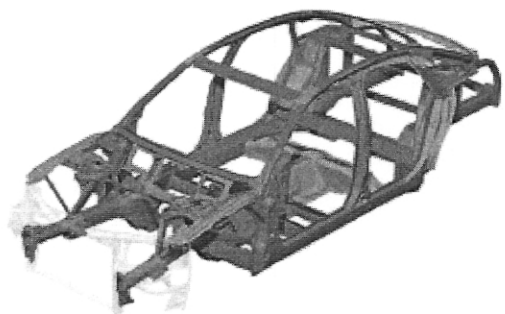
ภาพที่ 2.11 ข้อมูลการพยากรณ์อากาศบนลีนุกส์คลัสเตอร์ [10]

2) ด้านการคำนวณผลภาพถ่ายดาวเทียม ในการวางผังเมือง การจัดการที่ดิน การจัดการระบบจราจร การวางแผนป้องกันน้ำท่วมจำเป็นต้องใช้ภาพถ่ายดาวเทียมหรือภาพถ่ายทางอากาศเพื่อมาวิเคราะห์หาข้อมูลต่าง ๆ ที่จำเป็น ซึ่งการใช้ซอฟต์แวร์เพื่อวิเคราะห์ภาพถ่ายดาวเทียมดังกล่าวใช้ระยะเวลาในการทำงานค่อนข้างนาน ลีนุกส์คลัสเตอร์จึงเป็นอีกทางเลือกหนึ่งที่มีผู้นิยมใช้มาวิเคราะห์ผลภาพถ่ายดาวเทียมภาพถ่ายทางอากาศซอฟต์แวร์ที่ใช้ในการคำนวณผลภาพถ่ายดาวเทียม เช่น GRASS (Geographic Resources Analysis Support System) ของ GRASS Research Group มหาวิทยาลัย Baylor ประเทศสหรัฐอเมริกา

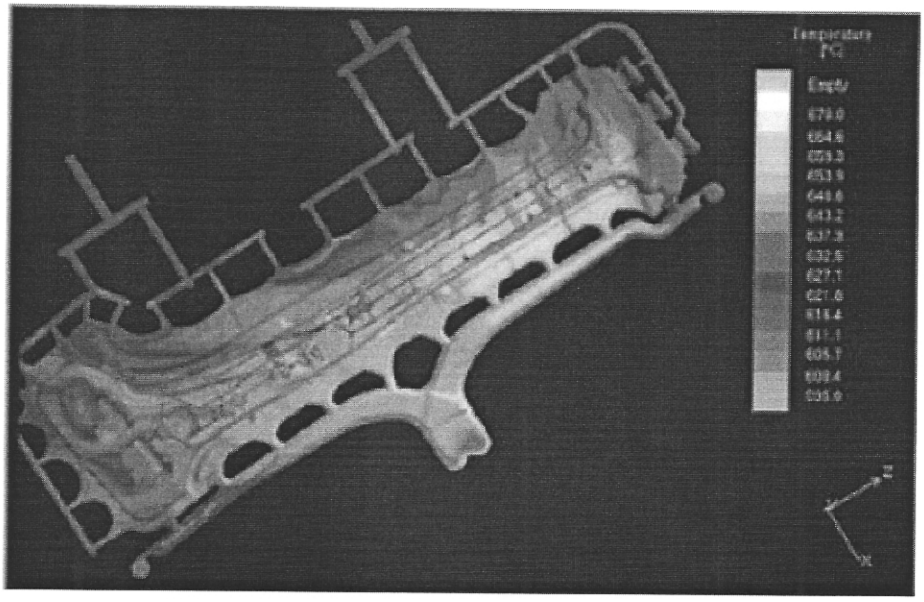


ภาพที่ 2.12 ภาพถ่ายทางดาวเทียมจังหวัดอุบลราชธานี [10]

3) ด้านการออกแบบเครื่องมือทางวิทยาศาสตร์และยานพาหนะ การออกแบบเครื่องมือทางวิทยาศาสตร์ที่สำคัญ เช่น การออกแบบกล้องโทรทรรศน์ขนาดใหญ่การออกแบบเครื่องมือที่ใช้ในห้องปฏิบัติการวิจัยรวมถึงการออกแบบยานพาหนะที่ใช้กันในปัจจุบันนิยมสร้างแบบจำลองโดยใช้คอมพิวเตอร์ก่อนที่จะสร้างจริงเพื่อที่จะได้สามารถตรวจสอบข้อผิดพลาดต่าง ๆ ก่อนที่จะสร้างจริงการรันโปรแกรมที่ใช้ในการออกแบบ (CAD/CAM) ต่าง ๆ ค่อนข้างจะใช้ระยะเวลานาน ดังนั้นจึงมีผู้ผลิตอุปกรณ์และยานพาหนะบางรายนำเทคโนโลยีลีนุกส์คลัสเตอร์ไปประยุกต์ใช้

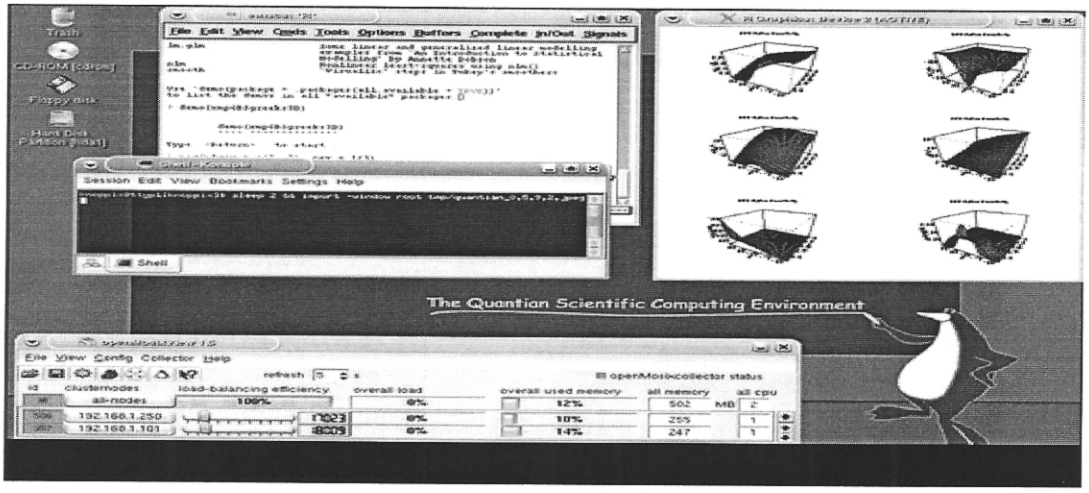


ภาพที่ 2.13 การออกแบบโครงสร้างรถยนต์ [11]



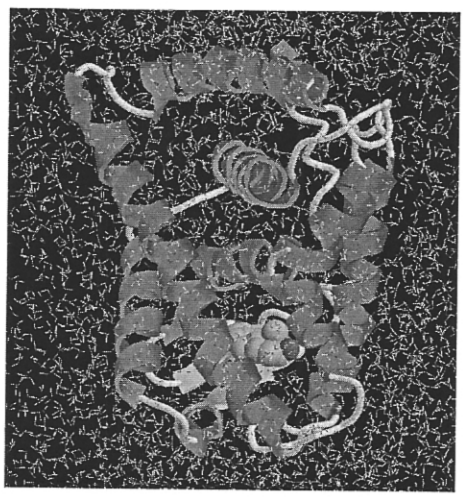
ภาพที่ 2.14 โปรแกรมจำลองการไหลของอะลูมิเนียมของบริษัท Audi บนลีนุกส์คลัสเตอร์ [11]

4) ด้านการจำลองแบบทางฟิสิกส์ เทคโนโลยีทางด้านฟิสิกส์ซึ่งเป็นพื้นฐานต่าง ๆ ทางด้านวิศวกรรมในปัจจุบันได้รุดหน้าไปมากมีผู้หันมานิยมใช้การจำลองแบบในงานที่เกี่ยวข้องกับฟิสิกส์มากขึ้น เช่น การจำลองรูปแบบการกระจายความร้อน การจำลองรูปแบบทางด้านนิวเคลียร์ฟิสิกส์ แต่อย่างไรก็ตามโปรแกรมจำลองด้านฟิสิกส์ จำเป็นต้องใช้เครื่องที่มีประสิทธิภาพในการคำนวณสูง ดังนั้นจึงมีการประยุกต์ใช้กับลีนุกส์คลัสเตอร์มากขึ้นเรื่อย ๆ



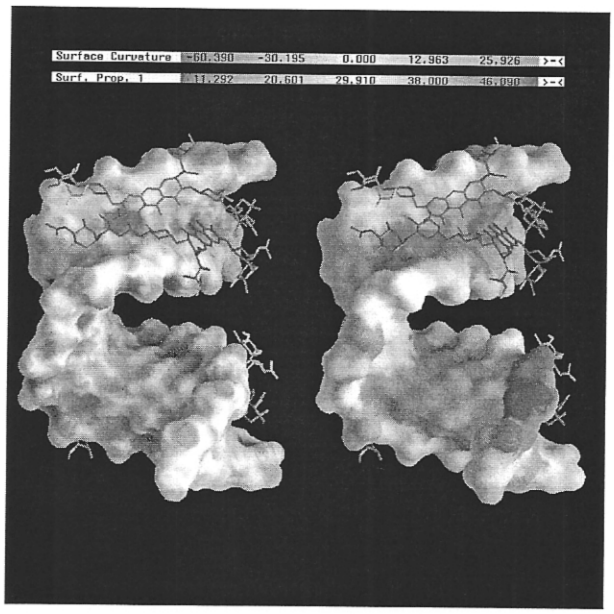
ภาพที่ 2.15 แบบจำลองทางฟิสิกส์บนลีนุกส์คลัสเตอร์ [12]

5) ด้านการจำลองแบบทางเคมี วิทยาการทางด้านเคมีเริ่มหันมาใช้การจำลองแบบมากขึ้น เช่น การจำลองแบบมลพิษในอากาศ การศึกษาพฤติกรรมของอนุภาคใน Fluidize-Based การคำนวณพฤติกรรมต่าง ๆ ของของไหลเป็นต้น ซึ่งการจำลองแบบเหล่านี้เป็นพื้นฐานที่สำคัญในโรงงานต่าง ๆ โปรแกรมการจำลองแบบดังกล่าวต้องอาศัยระยะเวลาในการคำนวณค่อนข้างนาน ดังนั้นลีนุกส์คลัสเตอร์จึงเป็นอีกทางเลือกหนึ่งของหน่วยงานต่าง ๆ เพื่อที่จะใช้ในการพัฒนาโปรแกรมจำลองแบบทางด้านเคมี



ภาพที่ 2.16 การจำลองทางด้านเคมีเกี่ยวกับกลุ่มโปรตีน [13]

6) ด้านการจำลองแบบทางชีววิทยา ในปัจจุบันเทคโนโลยีในด้านการจำลองแบบทางชีววิทยาก้าวหน้าขึ้นมากจนสามารถจำลองแบบ Sub-Cellular (อนุภาคที่มีขนาดระหว่าง μM จนถึง nM) ได้ซึ่งมีผู้นิยมใช้ซอฟต์แวร์การจำลองแบบเพื่อจำลอง DNA การจำลองเซลล์ของกล้ามเนื้อตลอดจนถึงการทำยาซึ่งซอฟต์แวร์ดังกล่าวจำเป็นต้องรันบนเครื่องที่มีสมรรถนะสูง ดังนั้นลีนุกส์คลัสเตอร์จึงเป็นอีกหนึ่งสำหรับนักชีววิทยาที่ต้องการรันโปรแกรมการจำลองแบบทางชีววิทยา



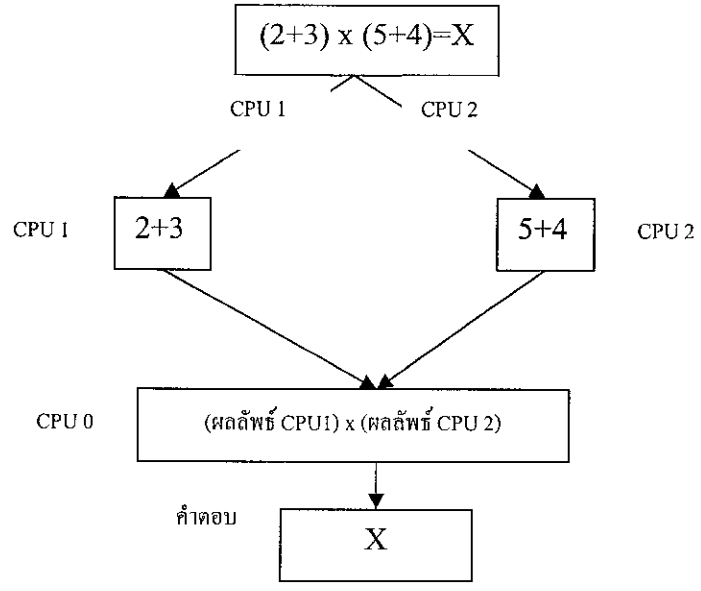
ภาพที่ 2.17 การจำลอง DNA [14]

2.1.6 การประมวลผลแบบขนาน (Parallel Computing)

แนวคิดการประมวลผลแบบขนานก็คือการที่มีปัญหาหนึ่งซึ่งมีขนาดใหญ่มากถ้าใช้หน่วยประมวลผลตัวเดียวก็จะประมวลผลได้ช้าแต่ถ้าจับงานนั้นมาแบ่งออกเป็นชิ้นเล็กหลาย ๆ ชิ้น แล้วกระจายงานเหล่านั้นให้หน่วยประมวลผลจำนวนมากช่วยกันประมวลผลก็จะทำให้งานนั้นเสร็จเร็วขึ้นเหมือนกับการสร้างตึกถ้าให้ช่างคนเดียวทำตึกหลังเล็ก ๆ ก็คงจะใช้นานมาก แต่ถ้ามีช่างหลายๆ คนช่วยกันก็จะทำให้ตึกนั้นเสร็จเร็วขึ้น [5]

ตัวอย่างงานทางด้านคอมพิวเตอร์ เช่นการคำนวณเพื่อหาค่า (2+3) x (5+4) = X ซึ่งถ้าใช้เพียงหน่วยประมวลผลเพียงตัวเดียวในการทำงานขั้นตอนในการประมวลผลก็จะเริ่มจากการหาบวกระหว่าง (2+3) ให้เสร็จก่อนแล้วจึงไปทำการบวกค่าระหว่าง (5+4) เมื่อได้คำตอบแล้วจึงจะนำเอาผลลัพธ์ที่ได้ระหว่าง (2+3), (5+4) มาทำการคูณกันเพื่อให้ได้มาซึ่งค่า X ซึ่งเป็นคำตอบ

ในขณะที่การประมวลผลแบบขนานจะมีหลักการทำงานที่แตกต่างไปโดยการประมวลผลแบบขนานนั้นจะมีแบ่งงานแล้วก็จะทำการกระจายงานนั้นไปยังหน่วยประมวลผลแต่ละตัวที่มีอยู่ในระบบซึ่งจะทำให้งานเสร็จได้เร็วขึ้น ดังภาพที่ 2.18



ภาพที่ 2.18 แสดงหลักการกระจายงานในแนวคิดแบบขนาน

จากภาพที่ 2.18 จะเห็นว่าจะมีการแบ่งงานและมีการกระจายงานต่าง ๆ ไปยังหน่วยประมวลผลต่าง ๆ ที่มีอยู่ในระบบโดยที่ CPU 1 ทำการบวกค่าระหว่าง 2 กับ 3 ไปพร้อม ๆ กันกับ CPU 2 ซึ่งจะทำหน้าที่บวกค่าระหว่าง 5 กับ 4 จากนั้นทั้งสอง CPU จะส่งผลลัพธ์ที่ได้ไปให้กับ CPU 0 ทำการคูณระหว่างผลลัพธ์ที่ได้จาก CPU 1 กับ CPU 2 ทำให้คำตอบของ X ซึ่งถ้าหากขนาดของข้อมูลมีขนาดใหญ่การประมวลผลแบบขนานจะทำงานได้เร็วกว่าการประมวลผลแบบหน่วยประมวลผลเดียวอย่างเห็นได้ชัด

2.2 MPI (Message Passing Interface)

2.2.1 MPI, Message Passing Interface

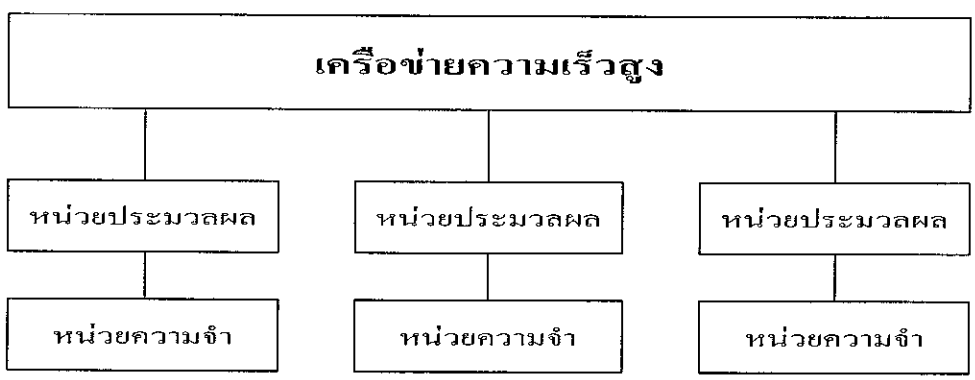
ในหนังสือ MPI-The Complete Reference Volume 1 [15] ได้ให้คำจัดความของ MPI ไว้ว่า MPI คือมาตรฐานที่ได้รับการออกแบบโดยคณะผู้ทำวิจัยจากวงการวิชาการและแวดวงอุตสาหกรรมถึงหน้าที่การทำงานที่หลากหลายของคอมพิวเตอร์ที่ทำงานแบบขนานการกำหนดมาตรฐานทางไวยากรณ์ และความหมายของหน้าที่หลัก ๆ ของไลบารี่ ที่เป็นประโยชน์ต่อผู้ใช้ทำให้สะดวกต่อการเขียนโปรแกรมแบบ message-passing ในภาษา Fortran, C, หรือ C++ โดย MPI เวอร์ชันที่ 1 ได้ทำการปล่อยออกมาในช่วงฤดูร้อนปี 1994 ตั้งแต่ MPI เวอร์ชันที่ 1 ปล่อยออกมา

MPI ได้ถูกกำหนดให้เป็นไลบารี่มาตรฐานสำหรับใช้ message-passing สำหรับคอมพิวเตอร์ที่ทำงานแบบขนาน

หรือกล่าวได้ว่า MPI (Message Passing Interface) ก็คือ Library มาตรฐานที่ใช้ในการเขียนโปรแกรมแบบขนาน ซึ่งทำงานร่วมกับภาษา Fortran, C, หรือ C++ โดย MPI เป็น Library ที่มีลักษณะเหมาะในการเขียนโปรแกรมเพื่อส่งข้อความแบบขนานที่ทำงานบนเครื่องกระจายหน่วยความจำ

2.2.2 การเขียนโปรแกรมแบบ Message Passing

การเขียนโปรแกรมด้วย MPI นั้น จะมองเหมือนกับว่าระบบคอมพิวเตอร์ที่ใช้อยู่เป็นคอมพิวเตอร์แบบขนานที่มีหน่วยความจำแยก หรือที่เรียกว่า "Distributed Memory Multicomputer" เป็นคำกล่าวของ ผศ.ดร. ภุชงค์ อุทโยภาศ [1] โดยมีโครงสร้างทั่วไป ดังภาพที่ 2.19



ภาพที่ 2.19 รูปแบบโครงสร้างคอมพิวเตอร์แบบขนานที่มีหน่วยความจำแยกจากกัน [1]

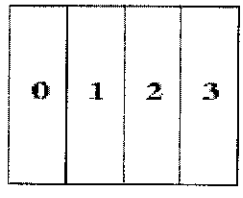
2.3 การกระจายข้อมูล (Data Scatter)

ระบบคลัสเตอร์ คือการนำเครื่องคอมพิวเตอร์แบบพีซีหลาย ๆ เครื่องมาทำการเชื่อมต่อกันบนระบบเครือข่ายความเร็วสูง (ความหมายของระบบคลัสเตอร์นั้นได้กล่าวไว้ในหัวข้อที่ 2.1.1) ซึ่งโครงสร้างการทำงานของระบบคลัสเตอร์นั้นจะอยู่ในรูปแบบของโครงสร้างคอมพิวเตอร์แบบขนานที่มีหน่วยความจำแยกจากกันโดยเด็ดขาด หมายความว่าเครื่องคอมพิวเตอร์แต่ละเครื่องจะมีหน่วยความจำสำหรับใช้งานเป็นของใครของมัน ดังภาพที่ 2.19 ซึ่งถ้าหากเครื่องคอมพิวเตอร์จำเป็นจะต้องมีการติดต่อสื่อสารกันก็จะใช้วิธีการแลกเปลี่ยน Message กันผ่านเครือข่ายคอมพิวเตอร์ ซึ่งการที่จะพัฒนาโปรแกรมแบบขนานบนระบบคลัสเตอร์ให้มีประสิทธิภาพในการทำงานจำเป็นต้อง

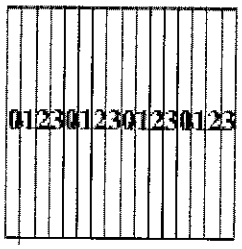
คำนึงถึง Communication Cost ที่เกิดขึ้นในระหว่างการประมวลผลซึ่งจะมีการส่งข้อมูลระหว่างเครื่องคอมพิวเตอร์แต่ละเครื่องบนระบบคลัสเตอร์หรือที่เรียกกันว่า โหนด (Node) ที่ทำการประมวลผลข้อมูลอยู่นั้นในบางครั้งการทำงานจะต้องมีการหยุดชะงักเนื่องจากการทำงานของแต่ละโหนดไม่มีความเป็นอิสระต่อกันเท่าที่ควรเพราะต้องรอรับข้อมูลจากโหนดอื่นเพื่อนำมาใช้ประมวลผลจึงจะสามารถทำการประมวลข้อมูลต่อไปได้ซึ่งการส่งข้อมูล (Send) และการรับข้อมูล (Receive) ระหว่างโหนดที่เกิดขึ้นในแต่ละครั้งจะทำให้เกิด Communication ตามมาด้วย James Demmel ได้กล่าวเอาไว้ว่า [4] ประสิทธิภาพในการทำงานของโปรแกรมที่พัฒนาจะดีหรือไม่ดีนั้นขึ้นอยู่กับจำนวนของ Communication ที่เกิดขึ้นในขณะที่มีการประมวลผล ถ้ามี Communication เกิดขึ้นมากประสิทธิภาพในการทำงานก็จะลดต่ำลง นอกจากนี้ James Demmel ยังกล่าวเอาไว้ว่าการเลือกใช้เครือข่ายที่มีประสิทธิภาพความเร็วในการทำงานสูงก็จะช่วยลดปัญหาของ Communication ที่เกิดขึ้นได้แต่จะต้องแลกกับการที่จะต้องจ่ายในราคาที่แพงจึงต้องมีประเมินถึงความคุ้มค่าที่ได้รับถ้าหากมีความจำเป็นจะต้องเลือกใช้เครือข่ายที่มีประสิทธิภาพความเร็วในการทำงานสูง

ดังนั้นการที่จะลดจำนวนของ Communication ที่เกิดขึ้นจำเป็นจะต้องมีการออกแบบขั้นตอนวิธีแบบขนาน (Parallel Algorithm) ให้มีประสิทธิภาพโดยต้องคำนึงถึงวิธีการกระจายข้อมูล (Scatter Data) ซึ่งการกระจายข้อมูล หมายถึงรูปแบบวิธีการที่ใช้แบ่งและกระจายข้อมูลไปให้แต่ละโหนดที่อยู่บนระบบคลัสเตอร์เพื่อช่วยกันประมวลผลข้อมูล และลดปัญหาต่าง ๆ ที่เป็นสาเหตุที่ทำให้เกิด Communication ซึ่งได้กล่าวไว้ในตอนต้นของหัวข้อนี้ ซึ่งการเลือกรูปแบบวิธีการกระจายข้อมูลที่เหมาะสมมาใช้เพื่อออกแบบขั้นตอนการประมวลผลแบบขนานจะช่วยเพิ่มประสิทธิภาพในการทำงานได้มาก

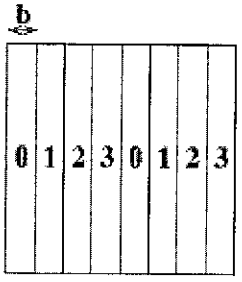
การจัดรูปแบบข้อมูลแบบ Column Block [3], Column Cyclic [3], Column Block Cyclic [3], Row and Column Block Cyclic [3] และ Block Skewed [3] ได้ถูกนำมาประยุกต์ใช้กับกับขั้นตอนวิธีการแบบขนานเพื่อแก้ปัญหาสมการเชิงเส้น Ax=b ด้วยวิธี Gaussian Elimination ดังรูปต่อไปนี้



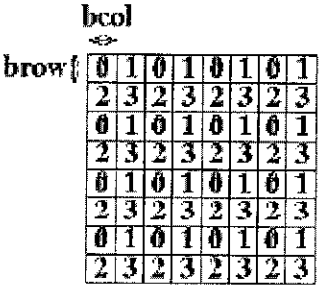
ภาพที่ 2.20 การจัดรูปแบบข้อมูลแบบ Column Block



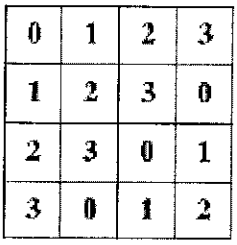
ภาพที่ 2.21 การจัดรูปแบบข้อมูลแบบ Column Cyclic



ภาพที่ 2.22 การจัดรูปแบบข้อมูลแบบ Column Block Cyclic



ภาพที่ 2.23 การจัดรูปแบบข้อมูลแบบ Row and Column Block Cyclic



ภาพที่ 2.24 การจัดรูปแบบข้อมูลแบบ Block Skewed

2.4 การแก้สมการเชิงเส้นด้วยวิธี Crout Decomposition

ก่อนที่จะทราบถึงวิธีการแก้สมการ Ax=b ด้วยวิธี Crout Decomposition นั้นจะกล่าวถึงความรู้พื้นฐานในเรื่องของเมตริกซ์ (Matrix) ก่อนเนื่องจากวิธีการแก้สมการจะต้องอาศัยความรู้ทางด้านเมตริกซ์เข้ามาช่วยในการแก้ปัญหาสมการเชิงเส้น จากนั้นก็จะกล่าวถึงวิธีการแก้ปัญหาสมการเชิงเส้นด้วยวิธีการต่างๆ และหัวข้อสุดท้ายก็จะกล่าวถึงวิธีการแก้สมการเชิงเส้นด้วยวิธีCrout Decomposition

2.4.1 เมตริกซ์ (Matrix)

เมตริกซ์ [7, 9, 16] คือ อาร์เรย์ (Array) ของเลขจำนวนเต็ม, เลขจำนวนจริง, Operators หรือ Functions โดยจัดอยู่ในกรอบวงเล็บใหญ่ [ ] สมาชิกที่เรียงในแถวของ Array เรียกว่า แถว (Row) สมาชิกที่เรียงในหลักของ Array เรียกว่า หลัก (Column) โดยทั่ว ๆ ไปสัญลักษณ์ของเมตริกซ์จะแทนด้วยอักษรตัวพิมพ์ใหญ่ของภาษาอังกฤษ เช่น A, B, C เป็นต้น สมาชิกของเมตริกซ์จะแทนด้วยตัวพิมพ์เล็กและมีตัวห้อย i และ j ตัว i จะแทนตำแหน่งแถว ตัว j จะแทนตำแหน่งหลักของสมาชิกในเมตริกซ์

i จะมีค่า 1, 2, 3, ...m

j จะมีค่า 1, 2, 3, ...n

เช่น
$$A=\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}_{mxn}$$

a_{11} แสดงว่าเป็นสมาชิกที่อยู่ที่ตำแหน่ง แถว ที่ 1 หลัก ที่ 1

a_{23} แสดงว่าเป็นสมาชิกที่อยู่ที่ตำแหน่ง แถว ที่ 2 หลัก ที่ 3

a_{31} แสดงว่าเป็นสมาชิกที่อยู่ที่ตำแหน่ง แถว ที่ 3 หลัก ที่ 1

ชนิดของ Matrices

1. เมตริกซ์แถว (Row Matrix) คือเมตริกซ์ที่มีเพียงแถวเดียว เช่น $\begin{bmatrix}1 & 5 & 3\end{bmatrix}, \begin{bmatrix}2 & 8\end{bmatrix}$
2. เมตริกซ์หลัก (Column Matrix) คือเมตริกซ์ที่มีเพียงหลักเดียว เช่น $\begin{bmatrix}4\\5\\9\end{bmatrix}, \begin{bmatrix}2\\4\\6\\8\end{bmatrix}$
3. เมตริกซ์จัตุรัส (Square matrix) คือเมตริกซ์ที่มีจำนวน แถว และ หลัก เท่ากัน
4. เมตริกซ์ศูนย์ (Zero matrix) คือเมตริกซ์จัตุรัสที่สมาชิกทุกตัวมีค่าเป็นศูนย์หรือ $a_{ij}=0$ ทุก ๆ i และ j
5. เมตริกซ์หนึ่ง (Unit matrix) คือเมตริกซ์จัตุรัสที่สมาชิกทุกตัวมีค่าเป็นหนึ่งหรือ $a_{ij}=1$ ทุก ๆ i และ j
6. เมตริกซ์เอกลักษณ์ (Identity matrix) คือเมตริกซ์จัตุรัสที่สมาชิกตามแนวเส้นทแยงมุมหลัก (Main diagonal) มีค่าเป็น 1 ทุกตัว โดยทั่ว ๆ ไปจะใช้สัญลักษณ์ I นั่นคือ $I = a_{ij} = \begin{cases}1 & \forall \quad i=j\\0 & \forall \quad i\neq j\end{cases}$

เช่น $I=\begin{bmatrix}1 & 0 & 0\\0 & 1 & 0\\0 & 0 & 1\end{bmatrix}$ คุณสมบัติของ Identity IA = AI = A : Identity matrix คูณ กับ เมตริกซ์ใด ๆ จะไม่ทำให้เมตริกซ์ นั้นเปลี่ยนแปลง

7. เมตริกซ์สามเหลี่ยมบน (Upper Triangular Matrix) คือเมตริกซ์จัตุรัสที่สมาชิกที่อยู่ใต้แนวเส้นทแยงมุมหลักมีค่าเป็นศูนย์โดยทั่ว ๆ ไปจะใช้สัญลักษณ์ U นั่นคือ $U = a_{ij} = 0$ ทุก ๆ $i > j$

เช่น $U=\begin{bmatrix}1 & 2 & 6\\0 & 3 & 4\\0 & 0 & 1\end{bmatrix}$

8. เมตริกซ์สามเหลี่ยมล่าง (Lower Triangular Matrix) คือเมตริกซ์จัตุรัสที่สมาชิกที่อยู่เหนือแนวเส้นทแยงมุมหลักมีค่าเป็นศูนย์ โดยทั่ว ๆ ไปจะใช้ สัญลักษณ์ L นั่นคือ $L = a_{ij} = 0$ ทุก ๆ $i > j$ เช่น $L=\begin{bmatrix}1 & 0 & 0\\5 & 3 & 0\\2 & 2 & 1\end{bmatrix}$

9. เมตริกซ์เส้นทแยงมุม (Diagonal Matrix) คือเมตริกซ์จัตุรัสที่สมาชิกทุกตัวที่ไม่อยู่ในแนวเส้นทแยงมุมหลักมีค่าเป็นศูนย์ทั้งหมด สมาชิกที่อยู่ในแนวเส้นทแยงมุมหลักมีค่าเป็นศูนย์ได้แต่เป็นได้ไม่ทุกตัว โดยทั่ว ๆ ไปจะใช้สัญลักษณ์ D หรือ diag(A) นั่นคือ D หรือ diag(A) = 0 ทุก ๆ $i \neq j$ เช่น $diag(A) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

หมายเหตุ $\forall$ คือเครื่องหมาย For all หรือ ทุก ๆ หรือ ทั้งหมด

2.4.2 เวคเตอร์ (Vectors)

เวคเตอร์ [7, 17] คือการจัดเรียงจำนวนใด ๆ ให้อยู่ในรูปของเมตริกซ์ที่มีแถวเดียวหรือมีหลักเดียว เช่น

$\begin{bmatrix} 1 & 5 & 3 \end{bmatrix}, \begin{bmatrix} 2 & 8 \end{bmatrix}$ เรียกว่าเมตริกซ์ที่มีแถวเดียวว่า row vector

$\begin{bmatrix} 4 \\ 5 \\ 9 \end{bmatrix}, \begin{bmatrix} 2 \\ 4 \\ 6 \\ 8 \end{bmatrix}$ เรียกว่า เมตริกซ์ที่มีหลักเดียวว่า column vector

2.4.3 ทรานสโฟสของเมตริกซ์ (Transposition of Matrix)

ทรานสโฟสของเมตริกซ์ [7, 17] คือวิธีการสลับตำแหน่งให้แถวเป็นหลักและให้หลักเป็นแถวใช้สัญลักษณ์ A^T

คุณสมบัติการทรานสโฟส

1. ถ้า A มีมิติเท่ากับ mxn จะพบว่า A^T จะมีมิติเท่า nxm
2. ถ้า A = a_{ij} แล้ว $A^T = a_{ji}$
3. $\left[A^T\right]^T = A$

ตัวอย่าง การทรานสโฟสของเมตริกซ์

$$A = \begin{bmatrix} 1 & -2 & -5 \\ 8 & 3 & 6 \\ 9 & 8 & -1 \end{bmatrix}_{3x3}$$

$$A^T = \begin{bmatrix} 1 & 8 & -5 \\ -2 & 3 & 8 \\ -5 & 6 & -1 \end{bmatrix}_{3x3}$$

$$[A^T]^T = \begin{bmatrix} 1 & -2 & -5 \\ 8 & 3 & 6 \\ 9 & 8 & -1 \end{bmatrix}_{3x3}$$

ข้อสังเกต การทรานสโฟสจะเป็นเพียงการสลับตำแหน่งของสมาชิกในกลุ่มสามเหลี่ยมบนกับกลุ่มสามเหลี่ยมล่างเท่านั้น สมาชิกที่ในแนวเส้นทแยงมุมหลักจะไม่เปลี่ยนแปลงตำแหน่ง

2.4.4 การบวก ลบ เมตริกซ์

เมตริกซ์ที่จะสามารถบวกลบกันได้นั้นจะต้องมีมิติเท่ากัน คือมีจำนวนแถวเท่ากันและมีจำนวนหลักเท่ากันผลที่ได้จะมีมิติเท่ากัน $C = A_{ij} + B_{ij}$ สำหรับ i = k และ j = 1 การบวกลบกันให้นำเอาสมาชิกที่อยู่ในตำแหน่งเดียวกันทำการบวกลบกัน

ตัวอย่าง ของการบวกและลบเมตริกซ์

$$A = \begin{bmatrix} 1 & 8 & -3 \\ 2 & 3 & 8 \\ -5 & 6 & 4 \end{bmatrix}_{3x3}, \quad B = \begin{bmatrix} 1 & -2 & -5 \\ 8 & 3 & 6 \\ 9 & 8 & -1 \end{bmatrix}_{3x3}$$

$$A + B = \begin{bmatrix} 2 & 6 & -8 \\ 10 & 6 & 14 \\ 4 & 14 & 3 \end{bmatrix}_{3x3}$$

$$A - B = \begin{bmatrix} 0 & 10 & 2 \\ -6 & 0 & 2 \\ -14 & 2 & 5 \end{bmatrix}_{3x3}$$

2.4.5 การคูณเมตริกซ์

ถ้าA, B เป็นเมตริกซ์ใด ๆ ซึ่งสามารถจะนำมาหาผลคูณได้ต้องมีคุณสมบัติดังนี้

1. ถ้า A เป็นเมตริกซ์ที่มีขนาด m x p และ B เป็นเมตริกซ์ที่มีขนาด q x n ผลคูณ AB จะเกิดขึ้นได้เมื่อ p = q และ AB จะมีขนาด m x n

2. ถ้า AB = C สมาชิกของผลคูณในแต่ละตำแหน่งจะหาได้โดยใช้หลักการดังนี้

$c_{ij} = a_{i1} b_{1j} + a_{i2} b_{2j} + a_{i3} b_{3j} + ... + a_{in} b_{nj}$ เมื่อ i, j แทนตำแหน่งของสมาชิกในเมตริกซ์

ตัวอย่าง การคูณเมตริกซ์

$$\begin{bmatrix}1 & 2\\ 3 & 4\\ 5 & 6\end{bmatrix}_{3x2} x \begin{bmatrix}1\\ 2\end{bmatrix}_{2x1} = \begin{bmatrix}1\cdot 1+ & 2\cdot 2\\ 3\cdot 1+ & 4\cdot 2\\ 5\cdot 1+ & 6\cdot 2\end{bmatrix} = \begin{bmatrix}5\\ 11\\ 17\end{bmatrix}_{3x1}$$

2.4.6 การหาคำตอบของระบบสมการเชิงเส้น (System of Linear Equations) โดยใช้เมตริกซ์ผกผันจากสมการเมตริกซ์

$$Ax=b \qquad (2.1)$$

เมื่อ A คือเมตริกซ์สัมประสิทธิ์ของสมการเชิงเส้น

x คือเวคเตอร์ของตัวแปรอิสระ

b คือเวคเตอร์ของค่าคงที่ ๆ เป็นค่า input ของระบบ

สมการพีชคณิตเชิงเส้นซึ่งเขียนแทนระบบใด ๆ เขียนอยู่ในรูปทั่วไปได้ดังนี้

$$\begin{aligned}
a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n &= b_1\\
a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n &= b_2\\
&\vdots\\
a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n &= b_m
\end{aligned}$$

เมื่อ a_{ij} เป็นสัมประสิทธิ์ค่าคงที่ และ b_i เป็นค่าคงที่

เพื่อให้สามารถแก้ปัญหาสมการเชิงเส้นได้ง่ายขึ้นควรทำการจัดรูปสมการที่อยู่ในรูปทั่วไปให้อยู่ในรูปของเมตริกซ์ Ax=b ได้ดังนี้

1. นำสัมประสิทธิ์ค่าคงที่ที่อยู่ทางด้านซ้ายของเครื่องหมายเท่ากับไปเขียนให้อยู่ในรูปเมตริกซ์ โดยจะทำการเรียงลำดับของสมการจากซ้ายไปขวาและจากบนลงล่าง ถ้าหากตำแหน่งใดของสมการไม่ตัวแปรให้แทนค่าตำแหน่งนั้นด้วยค่า ศูนย์ และทำกำหนดให้เมตริกซ์นี้เป็นเมตริกซ์ A

2. นำค่าคงที่ซึ่งอยู่ทางด้านขวาของเครื่องหมายเท่ากับไปเขียนให้อยู่ในรูปของเวคเตอร์โดยจะทำการเรียงลำดับค่าคงที่จากบนลงล่างและกำหนดให้เวคเตอร์นี้เป็นเวคเตอร์ b

3. กำหนดให้ x คือเวคเตอร์ของตัวแปรอิสระที่ยังไม่ทราบค่า

สมการพีชคณิตเชิงเส้นเขียนให้อยู่ในรูปเมตริกซ์ Ax=b ได้ดังนี้

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ . \\ x_m \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ . \\ . \\ . \\ b_m \end{bmatrix}$$

ตัวอย่าง การเปลี่ยนรูปสมการเชิงเส้นจากรูปทั่วไปให้อยู่ในรูปเมตริกซ์ Ax=b

$$x_1 + 2x_2 + 2x_3 = 5$$
$$2x_1 + 3x_3 = 3$$
$$3x_1 + 3x_2 + 2x_3 = 6$$

เมื่อจัดให้อยู่ในรูปเมตริกซ์จะได้ $\underset{A}{\begin{bmatrix} 1 & 2 & 2 \\ 2 & 0 & 3 \\ 3 & 3 & 2 \end{bmatrix}} \underset{x}{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}} = \underset{b}{\begin{bmatrix} 5 \\ 3 \\ 6 \end{bmatrix}}$

การจัดสมการเชิงเส้นให้อยู่ในรูปเมตริกซ์ได้แล้วก็จะทำให้ง่ายต่อการแก้สมการเพื่อหาค่า x

วิธีการที่จะใช้แก้สมการเชิงเส้น Ax=b ซึ่งถูกจัดให้อยู่ในรูปของเมตริกซ์นั้นสามารถกระทำได้หลายวิธีเช่น การใช้ทฤษฎี Cramer's Rule [7, 9, 16], Gauss Elimination [7, 9, 16], Gauss-Jordan Elimination [7, 9, 16], Cholesky Decomposition [7, 9, 16] และ LU Decomposition ซึ่งจะเกี่ยวข้องกับงานวิจัยนี้

2.4.6.1 LU Decomposition [5, 7, 9, 16] คือการเขียนแทนเมตริกซ์จัตุรัส (Square matrix) ใด ๆ ด้วยผลคูณของเมตริกซ์ L และเมตริกซ์ U คือ

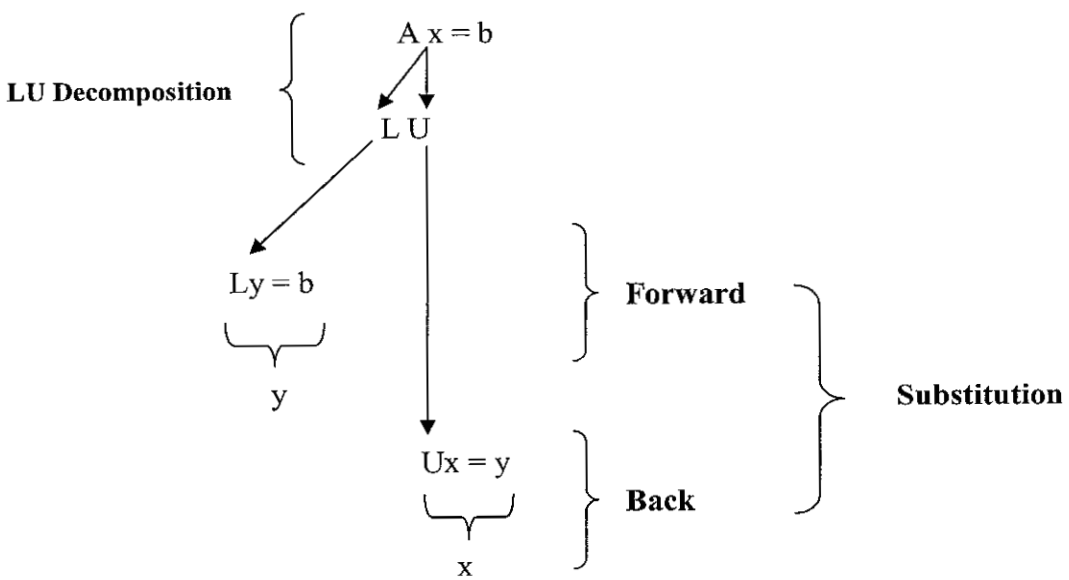
$$A = LU \tag{2.2}$$

เมื่อทราบค่าของเมตริกซ์ L และ U แล้วขั้นตอนต่อไปคือการนำเอาเมตริกซ์ L และ U ที่ได้ไปใช้เพื่อหาค่าของตัวแปรที่ยังไม่ทราบค่าโดยในที่นี้ก็คือค่า x ซึ่งสามารถหาได้จากสมการดังต่อไปนี้

$$L(Ux) = b \tag{2.3}$$

$$Ly = b \tag{2.4}$$

เมื่อ $$Ux = y \tag{2.5}$$



ภาพที่ 2.25 แสดงขั้นตอนการแก้สมการเชิงเส้นด้วยวิธี LU Decomposition [18]

โดยวิธีที่จะใช้ในการหา LU สามารถหาได้หลายวิธี เช่นกัน

1. Doolittle Decomposition หรือ Factorization [7, 9, 16] คือวิธีการหาเมตริกซ์ L และ U โดยใช้วิธีการ Gauss elimination โดยได้เมตริกซ์ L ที่มีสมาชิกในแนวเส้นทแยงมุมเป็น 1
2. Crout Decomposition [7, 9, 16] ซึ่งเป็นวิธีที่ใช้ในงานวิจัยนี้

2.4.6.2 Crout Decomposition [5, 7, 9, 16] คือวิธีแยกเมตริกซ์จัตุรัส (Square Matrix) A ให้อยู่ในรูปผลคูณของเมตริกซ์สามเหลี่ยมล่าง (Lower Triangular Matrix) L กับ เมตริกซ์สามเหลี่ยมบน (Upper Triangular Matrix) U โดยจะทำให้เมตริกซ์ U มีสมาชิกในแนวเส้นทแยงมุมเป็น 1

เมื่อเมตริกซ์ A มีขนาด nxn

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ . & . & \cdots & . \\ . & . & \cdots & . \\ . & . & \cdots & . \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}_{mxn}$$

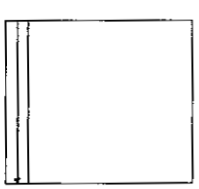
กำหนดให้ A = LU ดังนั้น

$$L = \begin{bmatrix} l_{11} & 0 & 0 & 0 & 0 \\ l_{21} & l_{22} & 0 & 0 & 0 \\ l_{31} & l_{32} & l_{33} & 0 & 0 \\ . & . & . & . & . \\ l_{n1} & . & . & . & l_{nn} \end{bmatrix}_{mxn} \qquad U = \begin{bmatrix} 1 & u_{12} & u_{13} & . & u_{1n} \\ 0 & 1 & u_{23} & . & u_{2n} \\ 0 & 0 & 1 & . & u_{nn} \\ . & . & . & . & . \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}_{mxn}$$

ขั้นตอนในการแยกเมตริกซ์จัตุรัส A ให้อยู่ในรูปผลคูณของ LU

1. หาค่าของสมาชิกหลักที่ 1 ของเมตริกซ์ L ซึ่งมีค่าเท่ากับสมาชิกตามตำแหน่งในหลักที่ 1 ของเมตริกซ์ A ดังภาพที่ 2.26

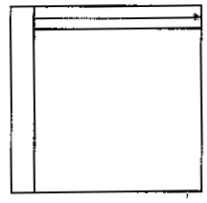
$$l_{i1} = a_{i1} \quad \text{สำหรับ } i = 1,2,...,n \tag{2.6}$$



ภาพที่ 2.26 รูปตำแหน่งสมาชิกของเมตริกซ์ L ที่หาได้จากสมการที่ 2.6 [16]

2. หาค่าของสมาชิกในตำแหน่งแถวที่ 1 หลักที่ 2 ถึงหลักที่ n ของเมตริกซ์ U ซึ่งมีค่าเท่ากับการนำเอาสมาชิกในตำแหน่งแถวที่ 1 หลักที่ 1 ของเมตริกซ์ A ไปหารตลอด ดังภาพที่ 2.27

$$u_{1j} = \frac{a_{1j}}{a_{11}} \quad \text{สำหรับ } j = 2,3,...,n \text{ เมื่อ } a_{11} \neq 0 \qquad (2.7)$$



ภาพที่ 2.27 รูปตำแหน่งสมาชิกของเมตริกซ์ U ที่หาได้จากสมการที่ 2.7 [16]

3. หาค่าของสมาชิกหลักที่ 2, 3,..., n ของเมตริกซ์ L จาก

$$l_{ij} = a_{ij} - \sum_{k=1}^{j-1} l_{ik}u_{kj} \quad \text{สำหรับ} \begin{cases} (j = 2,3,..,n) \\ (i = j, j+1,...,n) \end{cases} \qquad (2.8)$$

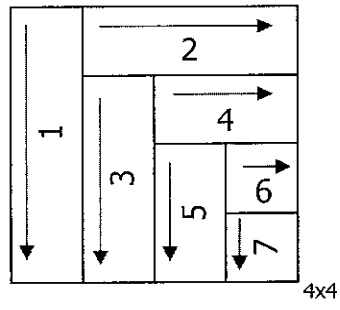
4. หาค่าของสมาชิกแถวที่ 2, 3,..., n ของเมตริกซ์ U จาก

$$u_{ik} = \frac{a_{jk} - \sum_{i=1}^{j-1} l_{ji}u_{ik}}{l_{jj}} \quad \text{สำหรับ} \begin{cases} (j = 2,3,..,n) \\ (k = j+1, j+2,...,n) \end{cases} \text{เมื่อ } l_{jj} \neq 0 \qquad (2.9)$$

หมายเหตุ การแยกเมตริกซ์ A ให้อยู่ในรูปผลคูณของ LU จากสมการที่ 2.6 - 2.9จะเริ่มต้นจากการหาสมาชิกในหลักที่ 1 ก่อน แล้วตามด้วยแถวที่ 1 จากนั้นจะสลับกลับมาหาในหลักที่ 2 แล้วตามด้วยแถวที่ 2 โดยที่จะทำการสลับหาค่าสมาชิกแบบนี้ไปจนกระทั่งถึงหลักที่ n

เมื่อ $A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ . & . & \dots & . \\ . & . & \dots & . \\ . & . & \dots & . \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix}_{mxn}$

ทำให้สรุปเป็นสมการได้ดังนี้ $C_{a_{11}}, R_{a_{21}}, C_{a_{22}}, R_{a_{23}}, ..., R_{a_{m-1n}}, C_{a_{mn}}$ เมื่อ C แทนหลัก และ R แทนแถว และ a ตำแหน่งสมาชิก



ภาพที่ 2.28 แสดงให้เห็นถึงลำดับขั้นตอนของการหาสมาชิกขนาด 4x4

ขั้นตอนการนำค่าของเมตริกซ์ L และ U ที่ได้มาใช้แก้สมการเพื่อหาค่า x ซึ่งเป็นตัวแปรที่ยังไม่ทราบค่านั้นสามารถหาค่า x ได้ 2 วิธี

1. ทำการแทนค่าเพื่อหาค่า x จากสมการ Ly=b และ Ux=y ดังนี้

   1.1 นำค่าของเมตริกซ์ L และ เวคเตอร์ b แทนค่าเพื่อทำการแก้สมการหาค่าของ y

   1.2 นำค่าของ y ที่ได้จากขั้นตอนที่ 1.1 มาแทนค่าลงในสมการ Ux=y เพื่อทำการหาค่า x ซึ่ง U ในที่นี้หมายถึงเมตริกซ์ U

2. ใช้วิธีการ Forward และ Back Substitution

   2.1 การหา Forward Substitution เป็นวิธีที่ใช้หาค่า y จากสมการ Ly=b ดังนี้

$$y_1 = \frac{b_{11}}{l_{11}} \quad \text{เมื่อ } l_{11} \neq 0 \tag{2.10}$$

$$y_i = \frac{1}{l_{ii}}\left(b_i - \sum_{j=1}^{i-1} l_{ij} y_j \right) \quad \text{สำหรับ i = 2,...,n เมื่อ n คือขนาดของเมตริกซ์ และ } l_{ii} \neq 0 \tag{2.11}$$

2.2 เมื่อได้ค่า y จาก Forward Substitution แล้วจะทำการหาค่า x จากสมการ Ux=y ด้วยวิธี Back Substitution ดังนี้

$$x_n = \frac{y_n}{u_{nn}} \text{ เมื่อ } u_{nn} \neq 0 \qquad (2.12)$$

$$x_i = \frac{1}{u_{ii}}\left(y_i - \sum_{j=i+1}^{n} u_{ij} x_j \right) \text{ สำหรับ i = n-1,...,1 เมื่อ n คือขนาดของเมตริกซ์และ } u_{ii} \neq 0 \qquad (2.13)$$

ตัวอย่าง ต่อไปนี้จะทำแก้สมการเชิงเส้นต่อไปนี้โดยใช้วิธี Crout Decomposition เพื่อหาคำตอบของตัวแปรที่ยังไม่ทราบค่า

$$x1 + 2x2 + x3 + 2x2 = 1$$
$$2x1 + x2 + 2x3 + x4 = 2$$
$$4x1 + 5x2 + x3 = 4$$
$$3x1 + 4x2 + 4x4 = 3$$

จาก Ax=B เพราะฉะนั้น $\begin{bmatrix} 1 & 2 & 1 & 2 \\ 2 & 1 & 2 & 1 \\ 4 & 5 & 1 & 0 \\ 3 & 4 & 0 & 4 \end{bmatrix} \begin{bmatrix} x1 \\ x2 \\ x3 \\ x4 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 4 \\ 3 \end{bmatrix}$

A x b

1. หา l_{i1} ได้จาก สมการที่ 1 (หาไปจนกว่าจะถึงแถวที่ 4)

$l_{11} = a_{11} = 1$

$l_{21} = a_{21} = 2$

$l_{31} = a_{31} = 4$

$l_{41} = a_{41} = 3$

2. หา u_{1j} = ได้จาก สมการที่ 2 (หาไปจนกว่าจะถึงหลักที่ 4)

$u_{12} = \frac{a_{12}}{l_{11}} = \frac{2}{1} = 2$

$$u_{13} = \frac{a_{13}}{l_{11}} = \frac{1}{1} = 1$$

$$u_{14} = \frac{a_{14}}{l_{11}} = \frac{2}{1} = 2$$

3. หา l_{ij} จากสมการที่ 3

For j = 2, i = 2

$$l_{22} = a_{22} - l_{21}u_{12} = 1-(2x2) = -3$$

For j = 2, i = 3

$$l_{23} = a_{23} - l_{31}u_{12} = 5-(4x2) = -3$$

For j = 2, i = 4

$$l_{42} = a_{42} - l_{41}u_{12} = 4-(3x2) = -2$$

4. หา u ที่เหลือ จากสมการที่ 4

For j = 2, k = 3

$$u_{23} = \frac{a_{23} - l_{21}u_{13}}{l_{22}} = \frac{2-(2x1)}{-3} = 0$$

For j = 2, k = 4

$$u_{24} = \frac{a_{24} - l_{21}u_{14}}{l_{22}} = \frac{1-(2x2)}{-3} = 1$$

For j = 3, k = 4

$$u_{34} = \frac{a_{34} - l_{31}u_{14} - l_{32}u_{24}}{l_{33}} = \frac{0-(4x2)-(-3x1)}{-3} = \frac{5}{3}$$

หมายเหตุ ข้อนี้ต้องต้องรอผลจาก l_{33} และ u_{24}

5. หา l ที่เหลือ จากสมการที่ 3

For j = 3, i = 3

$$l_{33} = a_{33} - \sum_{k=1}^{2} l_{3k}u_{k3} = a_{33} - l_{31}u_{13} - l_{32}u_{23}$$

$$= 1-(4x1)-(-3x0)$$

$$= -3$$

For j = 3, i = 4

$$l_{43} = a_{43} - \sum_{k=1}^{2} l_{4k}u_{k3} = a_{43} - l_{41}u_{13} - l_{42}u_{23}$$

$$= 0-(3x1)-(-2x0)$$

$$= -3$$

For j = 4, i = 4

$$l_{44} = a_{44} - \sum_{k=1}^{3} l_{4k}u_{k4} = a_{44} - l_{41}u_{14} - l_{42}u_{24} - l_{43}u_{34}$$

$$= 0-(3x2)-(-2x1)-(-3x\frac{5}{3})$$

$$= 5$$

เพราะฉะนั้น

$$U = \begin{bmatrix} 1 & 2 & 1 & 2 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 5/3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad L = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 2 & -3 & 0 & 0 \\ 4 & -3 & -3 & 0 \\ 3 & -2 & -3 & 5 \end{bmatrix}$$

จากนั้นแทนเพื่อหาค่า y จากสมการ Ly= b และใช้วิธีการ Forward Substitution เพื่อหาค่า y

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 2 & -3 & 0 & 0 \\ 4 & -3 & -3 & 0 \\ 3 & -2 & -3 & 5 \end{bmatrix} \begin{bmatrix} y1 \\ y2 \\ y3 \\ y4 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 4 \\ 3 \end{bmatrix}$$

$$y1 = \frac{1}{1}$$
$$y1 = 1$$

$$2y1 - 3y2 = 2$$
$$2(1) - 3y2 = 2$$
$$-3y2 = 2 - 2$$
$$y2 = \frac{0}{-3}$$
$$y2 = 0$$

$$4y1 - 3y2 - 3y3 = 4$$
$$4(1) - 3(0) - 3y3 = 4$$
$$-3y3 = 4 - 4$$
$$y3 = \frac{0}{-3}$$
$$y3 = 0$$

$$3y1 - 2y2 - 3y3 + 5y4 = 3$$
$$3(1) - 2(0) - 3(0) + 5y4 = 3$$
$$5y4 = 3 - 3$$
$$y4 = \frac{0}{5}$$
$$y4 = 0$$

$$\therefore y1 = 1, y2 = 0, y3 = 0, y4 = 0$$

นำค่า y ที่ได้มาทำการแทนค่าเพื่อหาค่า x ในสมการ Ux=y โดยใช้วิธี Back Substitution

$$\begin{bmatrix} 1 & 2 & 1 & 2 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 5/3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x1 \\ x2 \\ x3 \\ x4 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$x4 = \frac{0}{1}$$
$$x4 = 0$$

$$x3 + \frac{5}{3}x4 = 0$$
$$x3 = 0$$

$$x2 + x4 = 0$$
$$x2 = 0$$

$$x1 + 2x2 + x3 + 2x4 = 1$$
$$x1 = 1$$

ดังนั้น $x1 = 1, x2 = 0, x3 = 0, x4 = 0$

สรุปขั้นตอนในการแก้สมการ Ax=b ด้วยวิธี Crout Decomposition

1. แยกเมตริกซ์จัตุรัส A ให้อยู่ในรูปผลคูณของ LU (A=LU)
2. เมื่อได้ LU แล้วทำการหาค่า y จากสมการ Ly = b ด้วยวิธีการ Forward Substitution
3. เมื่อได้ค่า y แล้วทำการหาค่า x จากสมการ Ux = y ด้วยวิธีการ Back Substitution

หมายเหตุ การแก้สมการ Ly=b และ Ux=y ควรใช้วิธีการ Forward และ Back Substitution เนื่องจากจะเป็นวิธีที่ง่ายต่อการแทนค่าทั้งยังเป็นวิธีที่เหมาะสำหรับนำไปใช้ในการพัฒนาเป็นโปรแกรมคอมพิวเตอร์ต่อไป

2.5 งานวิจัยที่เกี่ยวข้อง

ในปัจจุบันนี้ได้มีหลาย ๆ งานวิจัยที่ได้ทำการศึกษาถึงประสิทธิภาพการทำงานของโปรแกรมแบบขนาน เช่น งานวิจัยของ จิรดา กันทรารักษ์ [19] เป็นงานวิจัยที่ได้ทำการเปรียบเทียบประสิทธิภาพในการประมวลผลแบบขนานบนระบบคลัสเตอร์ ระหว่างวิธีการแบบ Gaussian Elimination Pivoting กับวิธีการแบบ Gaussian Elimination Without Pivoting โดยทั้ง 2 วิธีเป็นส่วน

หนึ่งของวิธีการ Gaussian Elimination ซึ่งเป็นวิธีที่ใช้แก้ปัญหาสมการเชิงเส้นแต่สิ่งที่แตกต่างกันของทั้ง 2 วิธีนี้ก็คือ วิธีการแบบ Gaussian Elimination Pivoting จะใช้วิธีการสลับแถวเพื่อลดข้อผิดพลาดในการทำงาน ส่วนวิธีการแบบ Gaussian Elimination Without Pivoting จะไม่ใช้วิธีการสลับแถว โดยในงานวิจัยได้ใช้การจัดรูปแบบข้อมูลแบบ Column Block และแบบ Row Block เป็นการจัดรูปแบบข้อมูล ที่ใช้สำหรับการทดสอบในงานวิจัยเพื่อหาข้อสรุปว่าวิธีการใดใน 2 วิธีนี้มีประสิทธิภาพการทำงานมากกว่ากันเมื่อทำการประมวลผลแบบขนานบนระบบคลัสเตอร์

นอกจากนี้ก็ยังมีงานวิจัยของ Michael Wasilewski [20] และของ James Demmel [3] ซึ่งงานวิจัยของทั้งคู่ต้องการศึกษาว่าการจัดรูปแบบข้อมูลที่แตกต่างกันจะส่งผลต่อประสิทธิภาพการทำงานของโปรแกรมที่ประมวลผลแบบขนานหรือไม่ โดยงานวิจัยของ Michael Wasilewski ได้ทำการศึกษาและพัฒนาโปรแกรมประยุกต์เพื่อแก้ปัญหาสมการเชิงเส้นด้วยวิธีการ Gaussian Elimination เพื่อประมวลผลแบบขนานด้วยการจัดรูปแบบข้อมูลที่แตกต่างกัน 3 รูปแบบ คือ Column Block, Column Cyclic และ Column Block Cyclic โดยได้ทำการวิเคราะห์และเปรียบเทียบจากค่า Speedup และ Efficiency ที่เกิดขึ้นกับการจัดรูปแบบข้อมูลที่แตกต่างกันของทั้ง 3 รูปแบบ และจากผลการทดลองได้สรุปว่าการจัดรูปแบบข้อมูลมีผลต่อการประมวลแบบขนานซึ่งการจัดรูปแบบข้อมูลที่แตกต่างกันจะให้ประสิทธิภาพการทำงานที่แตกต่างกันด้วย โดยที่การจัดรูปแบบข้อมูลรูปแบบใดเกิด Communication น้อยลง รูปแบบนั้นก็จะให้ประสิทธิภาพการทำงานเพิ่มขึ้น

ในขณะที่งานวิจัยของ James Demmel [3] ได้ทำการทดสอบประสิทธิภาพการทำงานของการจัดรูปแบบข้อมูลที่แตกต่างกัน 5 รูปแบบ เมื่อนำมาใช้ประมวลผลแบบขนานเพื่อแก้ปัญหาสมการเชิงเส้นด้วยวิธีการ Gaussian Elimination ซึ่งการจัดรูปแบบข้อมูลทั้ง 5 รูปแบบที่ได้นำเสนอก็ได้แก่ Column Block,Column Cyclic,Column Block Cyclic,Row and Column Block Cyclic และ Block Skewed ซึ่งจากการทดลองได้สรุปว่าการจัดรูปแบบข้อมูลมีผลต่อประสิทธิภาพการทำงานของโปรแกรมแบบขนานโดยวิธีการจัดรูปแบบข้อมูลที่มีประสิทธิภาพมากที่สุดเมื่อนำไปใช้ประมวลผลแบบขนานเพื่อแก้ปัญหาสมการเชิงเส้นด้วยวิธีการ Gaussian Elimination ก็คือวิธีการจัดรูปแบบข้อมูลแบบ Row and Column Block Cyclic

อย่างไรก็ตามยังมีอีกหนึ่งงานวิจัยที่เป็นงานวิจัยของ Kaya, D. และ Wright, K [21] ซึ่งได้ทำการวิจัยเกี่ยวกับประสิทธิภาพความเร็วในการประมวลผลระหว่างการประมวลผลแบบขนานกับการประมวลผลแบบลำดับโดยใช้วิธี Crout Decomposition เป็นวิธีสำหรับใช้แก้ปัญหาสมการเชิงเส้น และใช้วิธีการจัดการข้อมูลแบบ Column Block สำหรับการประมวลผลแบบขนาน ซึ่งงานวิจัยได้แสดงให้เห็นว่าการประมวลผลแบบขนานบนระบบคลัสเตอร์ให้ประสิทธิภาพในการทำงานที่ดีกว่าการประมวลผลแบบลำดับ

ซึ่งจากงานวิจัยของ Michael Wasilewski และ James Demmel ได้แสดงให้เห็นว่าการจัดรูปแบบข้อมูล มีผลต่อประสิทธิภาพการทำงานของโปรแกรมแบบขนานแต่งานวิจัยส่วนใหญ่ได้ทำการศึกษาและวิจัยการจัดรูปแบบข้อมูลเพื่อนำไปประยุกต์ใช้กับการแก้ปัญหาสมการเชิงเส้นด้วยวิธีการ Gaussian Elimination เนื่องจากวิธีการ Gaussian Elimination เป็นวิธีที่ข้อมูลมีความเป็นอิสระต่อกันมากเมื่อนำไปใช้กับการประมวลผลแบบขนานทำให้ง่ายต่อการพัฒนาโปรแกรมประยุกต์แบบขนานบนระบบคลัสเตอร์ และมีประสิทธิภาพการในทำงานในขณะที่วิธีการแก้ปัญหาสมการเชิงเส้นด้วยวิธี Crout Decomposition เป็นวิธีที่ข้อมูลมีความเป็นอิสระต่อกันน้อยเมื่อนำไปใช้กับการประมวลผลแบบขนานทำให้มีการวิจัยและพัฒนาการจัดรูปแบบข้อมูลเพื่อนำไปประยุกต์ใช้กับการแก้ปัญหาสมการเชิงเส้นด้วยวิธี Crout Decomposition น้อยมาก เนื่องจากไม่มีประสิทธิภาพในการทำงาน และการที่จะนำวิธีการจัดรูปแบบข้อมูลที่ใช้กับวิธีการ Gaussian Elimination มาใช้กับวิธี Crout Decomposition นั้นอาจจะเป็นเรื่องที่ยากเนื่องจากมีความแตกต่างกันทางทฤษฎีของทั้ง 2 วิธี เพราะวิธีการแก้ปัญหาสมการเชิงเส้นด้วยวิธี Crout Decomposition ต้องใช้ผลลัพธ์ที่ได้ในแต่ละตำแหน่งช่วยหาผลลัพธ์ในตำแหน่งอื่น ๆ ของเมตริกซ์ต่อไปซึ่งทำให้ข้อมูลไม่เป็นอิสระต่อกันมากนักถ้านำไปใช้กับการประมวลผลแบบขนานจะทำให้มีจำนวน Communication เกิดขึ้นมาก ส่งผลในทางลบต่อประสิทธิภาพการทำงาน

ดังนั้นจึงทำให้เกิดแนวคิดที่จะออกแบบการจัดรูปแบบข้อมูลรูปแบบใหม่ที่ไม่ซ้ำกับรูปแบบเดิมที่มีอยู่แล้วขึ้นมาอีกหนึ่งรูปแบบซึ่งจะนำไปใช้แก้ไขปัญหาในเรื่องความไม่เป็นอิสระต่อกันของข้อมูลสำหรับการนำไปประยุกต์ใช้กับวิธีการแก้ปัญหาสมการเชิงเส้นด้วยวิธี Crout Decomposition เมื่อทำการประมวลผลแบบขนานบนระบบคลัสเตอร์โดยรูปแบบของการจัดรูปแบบข้อมูลแบบใหม่จะช่วยเพิ่มประสิทธิภาพในการทำงานสำหรับการประมวลผลแบบขนานบนระบบคลัสเตอร์ ให้กับวิธี Crout Decomposition เมื่อเปรียบเทียบกับการจัดรูปแบบข้อมูลแบบ Column Block และแบบ Row Block

บทที่ 3
การวิเคราะห์และออกแบบ

เนื้อหาในบทนี้จะกล่าวถึงการวิเคราะห์ประสิทธิภาพการทำงานระหว่างการประมวลผลแบบลำดับกับการประมวลผลแบบขนานโดยสิ่งที่จะเป็นตัวชี้วัดถึงประสิทธิภาพการทำงานของการประมวลผลแบบลำดับก็คือจำนวนของการคำนวณ (Computation) ที่เกิดขึ้นระหว่างการประมวลผลเท่านั้น ในขณะที่การประมวลผลแบบขนานประสิทธิภาพการทำงานจะขึ้นอยู่กับผลรวมของจำนวนของ Computation และจำนวนของการติดต่อสื่อสาร (Communication) ที่เกิดขึ้นในระหว่างการประมวลผลแต่ปัจจัยที่มีผลต่อประสิทธิภาพในการทำงานของการประมวลผลแบบขนานมากที่สุดก็คือจำนวนของ Communication ที่เกิดขึ้น [12, 22, 23]

เพราะฉะนั้นในงานวิจัยนี้จะทำการวิเคราะห์หาจำนวน Computation ที่เกิดขึ้นในขณะที่มีการประมวลผลแบบลำดับ และทำการวิเคราะห์หาจำนวนของ Computation และ Communication ที่เกิดขึ้นในขณะที่ประมวลผลแบบขนานเมื่อการประมวลผลทั้ง 2 แบบใช้การแก้ปัญหาสมการเชิงเส้น Ax=b ด้วยวิธี Crout Decomposition

จำนวนของ Computation คือการประมวลผลระหว่างนิพจน์ 2 นิพจน์ ด้วยเครื่องหมายทางด้านคณิตศาสตร์ (Operators) เช่น บวก, ลบ, คูณ, หาร ฯลฯ โดยเครื่องหมายทางคณิตศาสตร์ 1 เครื่องหมายจะนับเป็น 1 Computation [4, 20]

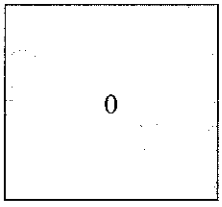
จำนวนของ Communication คือขณะที่มีการประมวลผลแบบขนานบนระบบคลัสเตอร์คอมพิวเตอร์แต่ละเครื่องจะมีการติดต่อสื่อสารกันเกิดขึ้นโดยการแลกเปลี่ยน Message กันผ่านเครือข่ายคอมพิวเตอร์ซึ่งการสื่อสารที่เกิดขึ้นแต่ละครั้งจะนับเป็น 1 Communication [4, 20]

3.1 การประมวลผลแบบลำดับ

การประมวลผลแบบลำดับคือ 1 เครื่องคอมพิวเตอร์จะมีเพียงหน่วยประมวลผลเดียวซึ่งสามารถประมวลผลงานได้เพียงครั้งละ 1 งานในหนึ่งหน่วยเวลาเท่านั้น โดยในการประมวลผลแบบลำดับนี้จะเกิดเฉพาะ Computation เพียงอย่างเดียวเนื่องจากการประมวลผลงานทุก ๆ งานจะกระทำบนเครื่องคอมพิวเตอร์เครื่องเดียวเท่านั้นจึงทำให้ไม่เกิดการติดต่อสื่อสารกันระหว่างคอมพิวเตอร์ทำให้ไม่มี Communication เกิดขึ้น

3.1.1 การวิเคราะห์ประสิทธิภาพการทำงาน

ต่อไปนี้จะทำการจำลองการคำนวณการประมวลผลแบบลำดับเพื่อที่จะทำให้ทราบว่ามีจำนวน Computation ที่เกิดขึ้นในการประมวลผลเท่าใด ถ้าทำการแก้ปัญหาสมการเชิงเส้นด้วยวิธี Crout Decomposition



ภาพที่ 3.1 Sequential Data Layout

1	5	6	7
2	8	11	12
3	9	13	15
4	10	14	16

ภาพที่ 3.2 ลำดับในการประมวลผลของ Sequential Data Layout

จากภาพที่ 3.1 ซึ่งเป็นรูป Data Layout ของการประมวลผลแบบลำดับ ซึ่งเลขศูนย์ที่อยู่ภายในหมายถึงเครื่องคอมพิวเตอร์ที่มีหน่วยประมวลผลเดียวซึ่ง Data Layout แบบ Sequential นี้สามารถอธิบายได้ว่า ไม่ว่าข้อมูลจะมีขนาดใหญ่เพียงใดก็จะมีเพียงหน่วยประมวลผลเดียวเท่านั้นที่จะทำหน้าที่ประมวลผล ขณะที่ภาพที่ 3.2 แสดงให้เห็นถึงลำดับในการประมวลผลเมื่อทำการแก้ปัญหาสมการเชิงเส้นด้วยวิธี Crout Decomposition โดยใช้รูปแบบของการประมวลผลเป็นแบบลำดับซึ่งจากรูปจะเห็นว่าเมตริกซ์นี้จะเป็นเมตริกซ์ที่มีขนาด 4x4 ซึ่งลำดับในการคำนวณนี้เกิดการมองว่าคอมพิวเตอร์ที่ประมวลผลแบบลำดับนี้ประมวลผลได้ทีละ 1 งานและในการประมวลผลก็ขึ้นอยู่กับวิธี Crout Decomposition (วิธี Crout Decomposition อ่านเพิ่มเติมได้ในบทที่ 2 หัวข้อที่ 2.4.6) ตารางที่ 3.1 จะเป็นการแสดงให้เห็นจำนวน Computation ของการประมวลผลแบบลำดับเมื่อทำการแก้ปัญหาสมการเชิงเส้นด้วยวิธี Crout Decomposition โดยในตารางมีขนาดของข้อมูลที่แตกต่างกัน ซึ่งทำให้เห็นความแตกต่างกันของจำนวน Computation ที่เกิดขึ้นเมื่อขนาดของข้อมูลเปลี่ยนไป

ตารางที่ 3.1 แสดงจำนวน Computation ของการประมวลผลแบบลำดับเพื่อแก้ปัญหาสมการเชิงเส้นด้วยวิธี Crout Decomposition

ขนาดของเมตริกซ์	จำนวน Computation
2x2	3
3x3	13
4x4	34
5x5	70
6x6	125
7x7	203
8x8	308

ดังนั้นในการหาค่าของจำนวน Computation ของการประมวลผลแบบลำดับด้วยวิธี Crout Decomposition จะเขียนอยู่ในรูป $\frac{2n^3}{3}$

3.2 การประมวลผลแบบขนาน

การประมวลผลแบบขนานคือการใช้หน่วยประมวลผลมากกว่าหนึ่งหน่วยประมวลช่วยกันประมวลผลข้อมูลที่มีอยู่ทั้งหมด โดยข้อมูลที่จะถูกแบ่งย่อยออกเป็นข้อมูลชิ้นเล็ก ๆ ก่อนแล้วจึงทำการกระจายข้อมูลเหล่านั้น ไปยังหน่วยประมวลผลที่มีอยู่ซึ่งในส่วนของรายละเอียดของการประมวลผลแบบขนานได้กล่าวเอาไว้ในบทที่ 2 หัวข้อที่ 2.1.6 แต่ในประมวลผลแบบขนานบนระบบคลัสเตอร์หน้าที่ในการแบ่งย่อยข้อมูล และการกระจายงานนั้นจะถูกกำหนดโดยผู้ใช้ซึ่งจะเป็นผู้ออกแบบและพัฒนาเป็นโปรแกรมเพื่อให้สามารถทำงานบนระบบคลัสเตอร์ได้

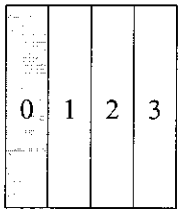
ดังนั้นก่อนที่จะเริ่มต้นทำการออกแบบการจัดรูปแบบข้อมูลแบบใหม่จึงควรที่จะทราบถึงประสิทธิภาพการทำงานของการจัดรูปแบบข้อมูลแบบ Column Block และแบบ Row Block ซึ่งเป็นเทคนิคพื้นฐานที่ใช้กระจายข้อมูล

3.2.1 การวิเคราะห์ประสิทธิภาพการทำงาน

ในการวัดประสิทธิภาพการทำงานของการประมวลผลแบบขนานนั้นก็จะใช้จำนวนของ Computation และจำนวนของ Communication ที่เกิดขึ้นในระหว่างการประมวลผลเป็นตัวชี้วัดถึงประสิทธิภาพซึ่งแตกต่างจากการประมวลผลแบบลำดับ เพราะการประมวลผลแบบลำดับจะคำนึงถึงเฉพาะจำนวนของ Computation ที่เกิดขึ้นเท่านั้นโดยในการนับจำนวนของ Computation และ Communication ที่เกิดขึ้นจะเป็นการนับจำนวนของ Computation และ Communication ที่จะสามารถเกิดขึ้นได้สูงสุด ซึ่งในความเป็นจริงอาจจะมีจำนวนที่น้อยกว่านี้

3.2.1.1 การจัดรูปแบบข้อมูลแบบ Column Block

การจัดรูปแบบข้อมูลแบบ Column Block [20] นี้จะทำการแบ่งย่อยข้อมูลให้มีขนาดเล็กตามแนวคอลัมน์จากนั้นก็จะทำการกระจายข้อมูลเหล่านั้นไปยังหน่วยประมวลผลต่าง ๆ ที่อยู่บนระบบคลัสเตอร์ โดยที่หน่วยประมวลแต่ละหน่วยบนระบบคลัสเตอร์จะถูกเรียกว่า โหนด โดยทั่วไปแล้วการแบ่งข้อมูลในรูปแบบนี้แต่ละโหนดจะได้รับข้อมูลที่เท่า ๆ กัน ดังภาพที่ 3.3 ซึ่งแสดงให้เห็นถึงการจัดรูปแบบข้อมูลแบบ Column Block โดยตัวเลขที่อยู่ภายในจะแทนด้วยชื่อของโหนด แต่ละโหนดบนระบบคลัสเตอร์ ซึ่งในรูปนี้จะมีโหนด ทั้งหมด 4 โหนดด้วยกัน



ภาพที่ 3.3 Column Block Data Layout

อย่างไรก็ตามในงานวิจัยนี้จะทำการทดสอบประสิทธิภาพการจัดรูปแบบข้อมูลโดยจะทำการแก้ปัญหาสมการเชิงเส้นด้วยวิธี Crout Decomposition ซึ่งข้อมูลที่จะนำมาทดสอบจะอยู่ในรูปของเมตริกซ์ ดังนั้นเพื่อทำให้เห็นถึงการจัดข้อมูลเมตริกซ์ด้วยเทคนิคการจัดรูปแบบข้อมูลแบบ Column Block จึงได้แสดงไว้ในภาพที่ 3.4 และ 3.5 โดยในภาพที่ 3.4 จะแสดงการจัดรูปแบบข้อมูลของเมตริกซ์ที่มีขนาด 4x4 ส่วนในภาพที่ 3.5 จะแสดงการจัดรูปแบบข้อมูลของเมตริกซ์ที่มีขนาด 8x8 สังเกตว่าเมตริกซ์ขนาด 8x8 แต่ละโหนดจะได้รับข้อมูล โหนดละ 2 คอลัมน์โดยวิธีที่ใช้ในการคำนวณหาจำนวนคอลัมน์ที่แต่ละโหนดจะได้รับในการจัดรูปแบบข้อมูลแบบ Column Block นั้น สามารถหาได้จากสมการที่ 3.1

$$\frac{n}{p} \tag{3.1}$$

เมื่อ n คือขนาดของเมตริกซ์

p คือจำนวนของโหนดทั้งหมดที่มีอยู่บนระบบคลัสเตอร์

0	1	2	3
0	1	2	3
0	1	2	3
0	1	2	3

ภาพที่ 3.4 การจัดรูปแบบข้อมูลเมตริกซ์ขนาด 4x4 แบบ Column Block

0	0	1	1	2	2	3	3
0	0	1	1	2	2	3	3
0	0	1	1	2	2	3	3
0	0	1	1	2	2	3	3
0	0	1	1	2	2	3	3
0	0	1	1	2	2	3	3
0	0	1	1	2	2	3	3
0	0	1	1	2	2	3	3

ภาพที่ 3.5 การจัดรูปแบบข้อมูลเมตริกซ์ขนาด 8x8 แบบ Column Block

ตารางที่ 3.2 แสดงจำนวน Computation และ Communication ของการประมวลผลแบบขนานด้วยการจัดรูปแบบข้อมูลแบบ Column Block เมื่อทำการแก้ปัญหาสมการเชิงเส้นด้วยวิธี Crout Decomposition

ขนาดของเมตริกซ์	จำนวน Computation	จำนวน Communication
2x2	3	2
3x3	12	8
4x4	29	20
5x5	56	35
6x6	99	57
7x7	164	84
8x8	253	108

ในตารางที่ 3.2 เป็นตารางที่แสดงจำนวน Computation และ Communication ของการประมวลผลแบบขนานด้วยการจัดรูปแบบข้อมูลแบบ Column Block เมื่อทำการแก้ปัญหาสมการเชิงเส้นด้วยวิธี Crout Decomposition โดยในตารางมีขนาดของข้อมูลที่แตกต่างกันซึ่งทำให้เห็นความแตกต่างกันของจำนวน Computation และ Communication ที่เกิดขึ้นเมื่อขนาดของข้อมูลเปลี่ยนไป (ตัวอย่างวิธีการคำนวณหา Computation และ Communication ดูได้จาก ภาคผนวก ก) ในการหาค่าของผลรวมระหว่างจำนวน Computation กับ Communication ของการจัดรูปแบบข้อมูลแบบ Column Block จะอยู่ในรูป $\frac{n^3}{3}+O(n^2)$

3.2.1.2 การจัดรูปแบบข้อมูลแบบ Row Block

การจัดรูปแบบข้อมูลแบบ Row Block [20] นี้จะทำการแบ่งย่อยข้อมูลตามแนวของแถวจากนั้นจะทำการกระจายข้อมูลไปยังหน่วยประมวลผลต่าง ๆ ที่อยู่บนระบบคลัสเตอร์ หรือที่เรียกว่า โหนด โดยการจัดรูปแบบข้อมูลแบบ Row Block แสดงได้ไว้ในภาพที่ 3.6 ซึ่งข้อมูลที่ได้ทำการแบ่งแล้วจะถูกกระจายไปยังโหนดต่าง ๆ ที่อยู่บนระบบคลัสเตอร์โดยในแต่ละโหนดจะได้รับข้อมูลเท่า ๆ กัน

นอกจากนี้ใน ภาพที่ 3.7 และภาพที่ 3.8 ก็เป็นการจัดรูปแบบข้อมูลของเมตริกซ์ด้วยการจัดรูปแบบข้อมูลแบบ Row Block โดยภาพที่ 3.7 เป็นเมตริกซ์ขนาด 4x4 ซึ่งแต่ละโหนดจะได้รับข้อมูลที่ถูกแบ่งแล้ว โหนดละ 1 แถว ในขณะที่ ภาพที่ 3.8 เป็นเมตริกซ์ขนาด 8x8 แต่ละโหนด

จะได้รับข้อมูลโหนดละ 2 แถว โดยวิธีที่ใช้ในการคำนวณหาจำนวนแถวที่แต่ละโหนดจะได้รับในการจัดรูปแบบข้อมูลแบบ Row Block นั้น สามารถหาได้จากสมการที่ 3.1

0
1
2
3

ภาพที่ 3.6 Row Block Data Layout

0	0	0	0
1	1	1	1
2	2	2	2
3	3	3	3

ภาพที่ 3.7 การจัดรูปแบบข้อมูลเมตริกซ์ขนาด 4x4 แบบ Row Block

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2
2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3
3	3	3	3	3	3	3	3

ภาพที่ 3.8 การจัดรูปแบบข้อมูลเมตริกซ์ขนาด 8x8 แบบ Row Block

ตารางที่ 3.3 แสดงจำนวน Computation และ Communication ของการประมวลผลแบบขนานด้วยการจัดรูปแบบข้อมูลแบบ Row Block เมื่อทำการแก้ปัญหาสมการเชิงเส้นด้วยวิธี Crout Decomposition

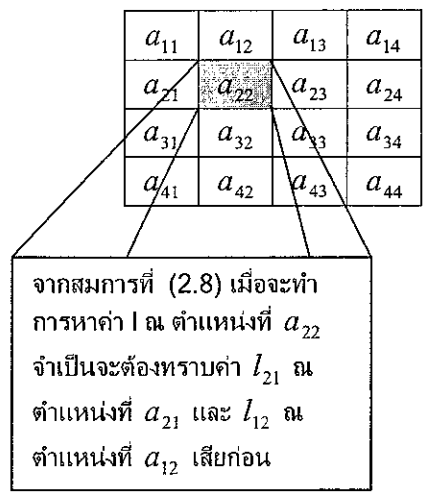
ขนาดของเมตริกซ์	จำนวน Computation	จำนวน Communication
2x2	3	1
3x3	11	5
4x4	26	14
5x5	50	26
6x6	91	44
7x7	153	66
8x8	238	90

ในตารางที่ 3.3 เป็นการแสดงจำนวน Computation และ Communication ของการประมวลผลแบบขนานด้วยการจัดรูปแบบข้อมูลแบบ Row Block เมื่อทำการแก้ปัญหาสมการเชิงเส้นด้วยวิธี Crout Decomposition โดยในตารางมีขนาดของข้อมูลที่แตกต่างกัน ซึ่งทำให้เห็นความแตกต่างกันของจำนวน Computation และ Communication ที่เกิดขึ้นเมื่อขนาดของข้อมูลเปลี่ยนไป (ตัวอย่างวิธีการคำนวณหา Computation และ Communication ดูได้จาก ภาคผนวก ก) โดยในการหาค่าของผลรวมระหว่างจำนวน Computation กับ Communication ของการจัดรูปแบบข้อมูลแบบ Row Block จะอยู่ในรูป $\frac{n^3}{3} + O(n^2)$

สังเกตว่าจำนวน Computation และ Communication ของการประมวลผลแบบขนานด้วยการจัดรูปแบบข้อมูลแบบ Row Block ในตารางที่ 3.3 จะมีจำนวนน้อยกว่า Computation และ Communication ของการประมวลผลแบบขนานด้วยการจัดรูปแบบข้อมูลแบบ Column Block ในตารางที่ 3.2 สาเหตุที่ทำให้การจัดรูปแบบข้อมูลแบบ Row Block มีจำนวน Computation และ Communication น้อยกว่าการจัดรูปแบบข้อมูลแบบ Column Block ก็เพราะลักษณะการคำนวณด้วยวิธี Crout Decomposition จะมีการคำนวณในแนวของคอลัมน์มากกว่าการคำนวณในแนวของแถว ซึ่งเมื่อมีการคำนวณหาค่าของเมตริกซ์ L และ เมตริกซ์ U จากเมตริกซ์ A เกิดขึ้น และอาจจะกล่าวได้ว่าการประมวลผลแบบขนานด้วยการจัดรูปแบบข้อมูลแบบ Row Block จะมีประสิทธิภาพมากกว่าการประมวลผลแบบขนานด้วยการจัดรูปแบบข้อมูลแบบ Column Block

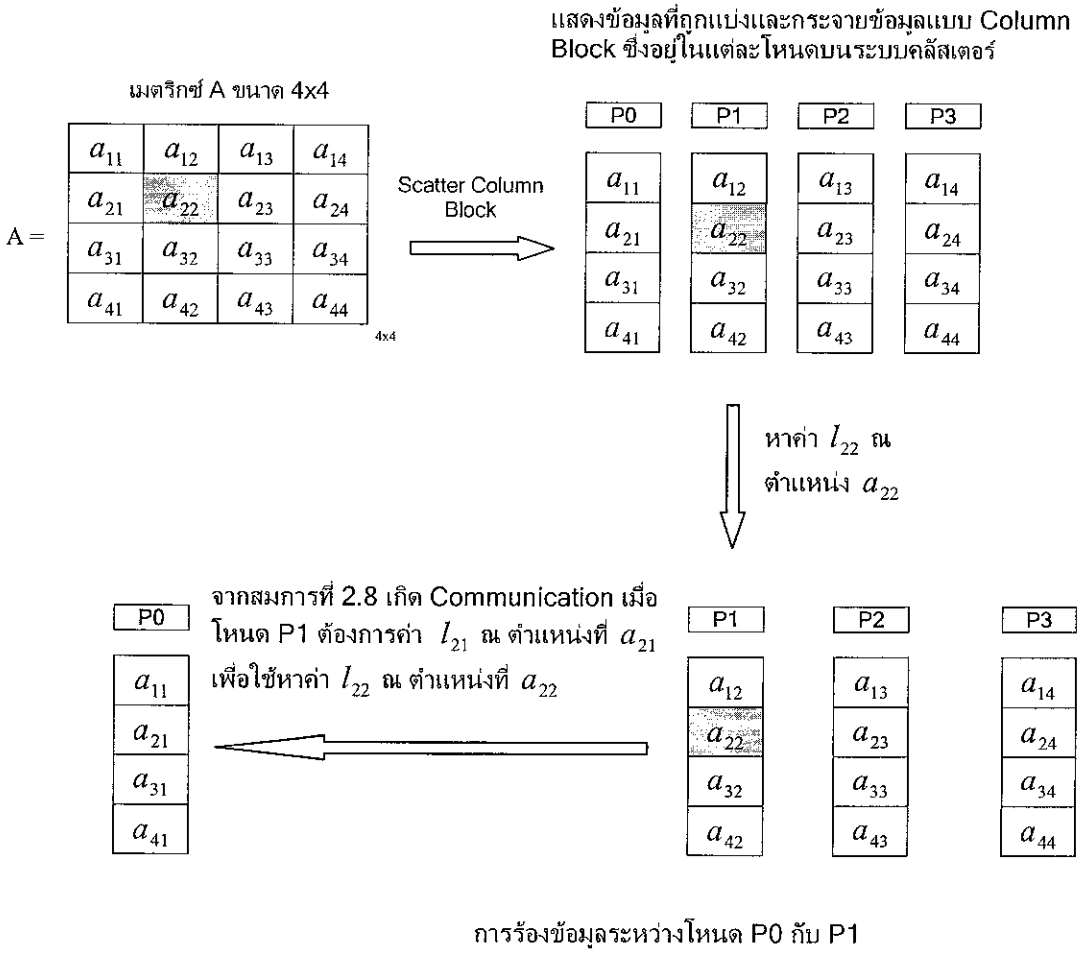
3.3 การออกแบบการจัดรูปแบบข้อมูลแบบ Right-angle Block

จากการจัดรูปแบบข้อมูลแบบ Column Block ในภาพที่ 3.3 และการจัดรูปแบบข้อมูลแบบ Row Block ในภาพที่ 3.6 นั้น การกระจายข้อมูลจะถูกแบ่งไปยังโหนดต่าง ๆ ในแบบคอลัมน์หรือแบบแถวเท่านั้น ซึ่งในบางครั้งเมื่อข้อมูลมีขนาดใหญ่อาจจะทำให้เกิดจำนวน Communication เพิ่มมากขึ้น แต่ถ้ามีการนำการจัดรูปแบบข้อมูลแบบ Column Block หรือ Row Block ไปใช้กับทฤษฎีทางคณิตศาสตร์, วิทยาศาสตร์ หรือ วิศวกรรมศาสตร์ บางทฤษฎี ซึ่งการเพิ่มมากขึ้นของจำนวน Communication ก็จะส่งผลต่อประสิทธิภาพของความเร็วในการประมวลผลแบบขนานบนระบบคลัสเตอร์ได้ [25] เพราะการประมวลผลแบบขนานจะมีหน่วยความจำแยกจากกันโดยเด็ดขาด หมายความว่าเครื่องคอมพิวเตอร์แต่ละเครื่องจะมีหน่วยความจำสำหรับใช้งานเป็นของใครของมัน ซึ่งถ้าหากเครื่องคอมพิวเตอร์จำเป็นจะต้องมีการติดต่อสื่อสารกัน ก็จะใช้วิธีการแลกเปลี่ยน Message กันผ่านเครือข่ายคอมพิวเตอร์ซึ่งการแลกเปลี่ยน Message ในแต่ละครั้งจะทำให้เกิด Communication และถ้ามีเกิด Communication มาก ๆ ก็จะทำให้ Traffic บนเครือข่ายที่ใช้ในการติดต่อสื่อสารติดขัด การทำงานจะต้องมีการหยุดชะงักเนื่องจากต้องรอการติดต่อสื่อสารจากคอมพิวเตอร์เครื่องอื่น ๆ บนเครือข่าย ทำให้ประสิทธิการทำงานไม่ดีเท่าที่ควรจะเป็น หรืออาจจะใช้เวลาในการประมวลผลนานกว่าการประมวลผลแบบลำดับ [20, 24] ก็เป็นได้ สาเหตุที่ทำให้การจัดรูปแบบข้อมูลแบบ Column Block และการจัดรูปแบบข้อมูลแบบ Row Block มีจำนวน Communication เพิ่มมากขึ้นก็เพราะเทคนิคการแบ่งและกระจายข้อมูลทั้ง 2 รูปแบบเหมาะสำหรับงานหรือทฤษฎีที่ข้อมูลไม่ค่อยมีความเกี่ยวพันธ์กันมากนัก หรือก็คือแต่ละโหนดสามารถที่จะประมวลผลข้อมูลในโหนดของตนได้โดยไม่จำเป็นต้องใช้ข้อมูล หรือรอข้อมูลจากโหนดอื่น ๆ หรือใช้น้อยที่สุด ซึ่งจะทำให้จำนวน Communication ลดลง เช่น งานที่เกี่ยวกับการเรนเดอร์ (Render) ภาพ หรือการใช้เพื่อประมวลทางงานด้านคณิตศาสตร์ ซึ่งการนำการจัดรูปแบบข้อมูลแบบ Column Block และแบบ Row Block มาใช้เพื่อแก้สมการเชิงเส้นด้วยวิธี Crout Decomposition แบบขนานบนระบบคลัสเตอร์อาจจะทำให้จำนวน Communication เพิ่มมากขึ้นกว่าปรกติ ทำให้ประสิทธิภาพของความเร็วในการประมวลไม่ดีเท่าที่ควรส่งผลต่อระยะเวลาที่ใช้ในการประมวลผลทำให้ต้องเวลามากขึ้นเพราะการแก้สมการเชิงเส้นด้วยวิธี Crout Decomposition นั้นข้อมูลจะไม่เป็นอิสระต่อกันเท่าไหร่นักเนื่องจากในบางครั้งการที่จะได้มาซึ่งผลลัพธ์ ณ ตำแหน่งใด ๆ ต้องอาศัยผลลัพธ์ที่ได้จากตำแหน่งที่อยู่ก่อนหน้ามาช่วยคำนวณ ซึ่งวิธี Crout Decomposition สามารถศึกษาได้จากบทที่ 2 หัวข้อที่ 2.4.6.1



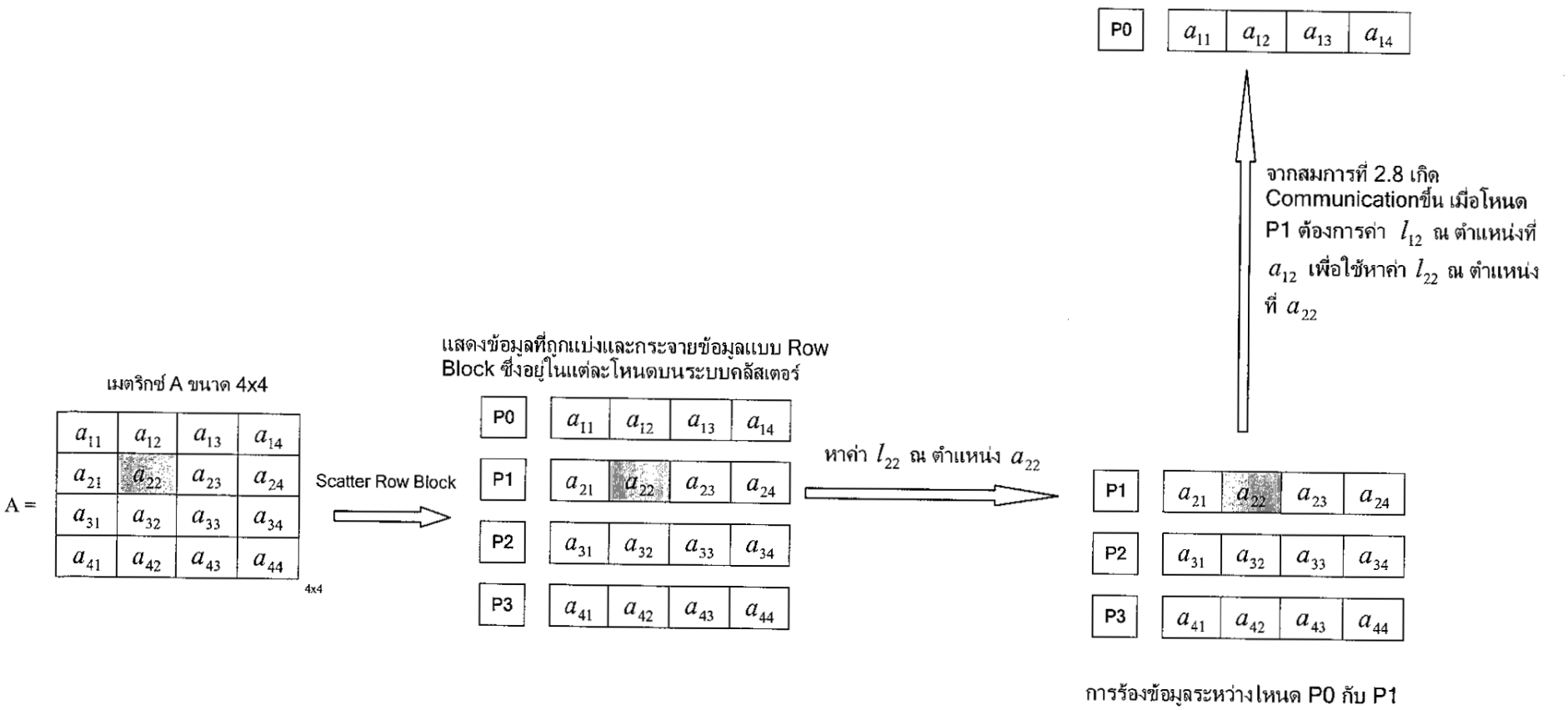
ภาพที่ 3.9 ความสัมพันธ์กันของข้อมูลเมื่อใช้วิธี Crout Decomposition แก้ปัญหา

ภาพที่ 3.9 แสดงให้เห็นว่าวิธีการแก้ปัญหาสมการเชิงเส้นด้วยวิธี Crout Decomposition นั้นข้อมูลจะไม่เป็นอิสระต่อกันมากนักโดยถ้านำเทคนิคการจัดรูปแบบข้อมูลแบบ Column Block หรือแบบ Row Block มาใช้ร่วมกับวิธีการแก้ปัญหาสมการเชิงเส้นด้วยวิธี Crout Decomposition ก็จะทำให้จำนวน Communication เพิ่มขึ้นซึ่งได้แสดงให้เห็นแล้วในตารางที่ 3.3 และ 3.2 ตามลำดับ



ภาพที่ 3.10 ตัวอย่าง Communication ที่เกิดขึ้นเมื่อใช้การจัดรูปแบบข้อมูลแบบ Column Block

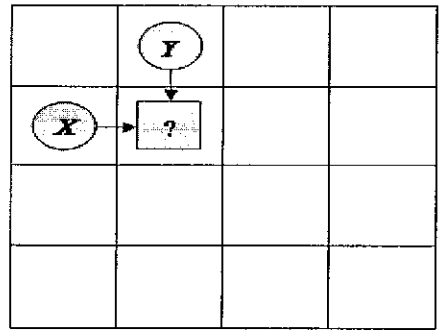
จากภาพที่ 3.10 แสดงให้เห็นตัวอย่างของ Comunication ที่เกิดขึ้นเมื่อจะแยกเมตริกซ์ A ให้อยู่ในรูป LU ด้วยวิธี Crout Decomposition ซึ่งทำการประมวลบนระบบคลัสเตอร์โดยใช้เทคนิคการจัดรูปแบบข้อมูลแบบ Column Block ขั้นแรกจะเริ่มจากการแบ่งเมตริกซ์ขนาด 4x4 แบบ Column Block ให้เป็นเมตริกซ์ย่อยขนาด 4x1 แล้วทำการกระจายข้อมูลเหล่านั้นไปยังโหนดทั้ง 4 โหนดที่อยู่บนระบบคลัสเตอร์โดยแต่ละโหนดจะมีชื่อ P0, P1, P2, P3 จากนั้นแต่ละโหนดจะเริ่มประมวลผลข้อมูลในส่วนที่ได้รับผิดชอบในขณะที่กำลังประมวลผลอาจจะต้องมีการติดต่อเพื่อขอข้อมูลจากโหนดอื่น ๆ ซึ่งจากภาพที่ 3.10 นั้นจะเห็นว่า โหนด P1 มีการร้องขอข้อมูล l_{21} จากโหนด P0 เพื่อที่จะใช้สำหรับหาค่า l_{22} โดยในภาพที่ 3.10 เป็นเพียงหนึ่งตัวอย่างของ Communication ที่เกิดขึ้น โดยในความเป็นจริง Communication ที่เกิดขึ้นจะมีมากกว่านี้ถ้าเมตริกซ์มีขนาดใหญ่



ภาพที่ 3.11 ตัวอย่างของ Communication ที่เกิดขึ้นเมื่อใช้การจัดรูปแบบข้อมูลแบบ Row Block

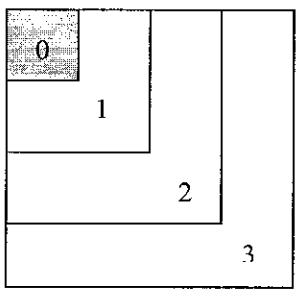
ในขณะที่ ภาพที่ 3.11 ก็เป็นอีกตัวอย่างหนึ่งที่แสดงให้เห็น Communication ที่เกิดขึ้นเมื่อจะทำการแยกเมตริกซ์ A ให้อยู่ในรูป LU ด้วยวิธี Crout Decomposition ซึ่งได้ทำการประมวลบนระบบคลัสเตอร์ แต่ใช้การจัดรูปแบบข้อมูลแบบ Row Block โดยในขั้นตอนแรกของการจัดรูปแบบข้อมูลด้วยรูปแบบนี้จะเริ่มจากการแบ่งเมตริกซ์ขนาด 4x4 ให้อยู่ในรูปแบบของ Row Block ทำให้ได้เมตริกซ์ย่อยที่มีขนาด 1x4 จากนั้นจึงทำการกระจายข้อมูลที่ได้ทำการแบ่งเรียบร้อยแล้วไปยังโหนดทั้ง 4 โหนด ที่อยู่บนระบบคลัสเตอร์ โดยแต่ละโหนดจะมีชื่อ P0, P1, P2, P3 เมื่อแต่ละโหนดได้รับข้อมูลแล้วก็จะเริ่มต้นการประมวลผลข้อมูลในส่วนที่ได้รับผิดชอบ จากภาพที่ 3.8 จะเห็นว่าขณะประมวลผลข้อมูล โหนด P1 ต้องมีการร้องขอข้อมูล l_{12} จากโหนด P0 เพื่อที่จะใช้สำหรับหาค่า l_{22} สังเกตว่ามี Communication เกิดขึ้นเหมือนกับการจัดรูปแบบข้อมูลแบบ Column Block ในภาพที่ 3.10 แต่ข้อมูลที่ทำการร้องข้อจากโหนด P0 นั้นจะเป็นคนละข้อมูลกันเนื่องจากใช้การจัดรูปแบบข้อมูลที่แตกต่างกัน ดังนั้นจำนวน Communication และจำนวน Computation ที่เกิดขึ้นก็จะแตกต่างกันด้วย ดูได้จากตารางที่ 3.2 และ 3.3

จากข้อบกพร่องที่เกิดขึ้นของการจัดรูปแบบข้อมูลแบบ Column Block และ แบบ Row Block ทำให้เกิดแนวคิดเกี่ยวกับการออกแบบการจัดรูปแบบข้อมูลแบบใหม่ที่จะช่วยลดจำนวน Computation และ Communication ที่เกิดขึ้นและจะส่งผลให้สามารถเพิ่มประสิทธิภาพของความเร็วที่ใช้ประมวลผลข้อมูลได้ ทำให้ช่วยลดเวลาที่ใช้ในการประมวลผลข้อมูลโดยการจัดรูปแบบข้อมูลแบบใหม่นี้จะใช้หลักการแบ่งและกระจายข้อมูลในรูปแบบที่คล้าย ๆ กับรูปมุมฉากแต่จะเป็นมุมฉากที่มีมุมอยู่ทางด้านขวา โดยแนวคิดนี้ได้มาจากการสังเกตวิธีการแยกเมตริกซ์ A ให้อยู่ในรูปผลคูณระหว่าง L กับ U ด้วยวิธี Crout Decomposition ซึ่งจากการสังเกตพบว่ามีขั้นตอนของการคำนวณเพื่อหาค่าของสมาชิกของเมตริกซ์ L และ U ดังภาพที่ 3.12 โดยในภาพแสดงให้เห็นว่าการที่จะได้มาซึ่งคำตอบที่อยู่ภายในช่องสี่เหลี่ยมนั้นจำเป็นจะต้องอาศัยผลลัพธ์ที่ได้จากคำตอบที่อยู่ภายในรูปวงกลมทั้ง 2 วงก่อน



ภาพที่ 3.12 โครงสร้างการคำนวณด้วยวิธี Crout Decomposition

จึงเรียกการจัดรูปแบบข้อมูลแบบใหม่ว่า Right-angle Block ดังแสดงในภาพที่ 3.13 ภาพที่ 3.14 และภาพที่ 3.15 แสดงตัวอย่างการจัดรูปแบบข้อมูลเมตริกซ์ด้วยเทคนิคการจัดรูปแบบข้อมูลแบบ Right-angle Block กับเมตริกซ์ขนาด 4x4 และ 8x8 ตามลำดับ



ภาพที่ 3.13 Right-angle Block Data Layout

0	1	2	3
1	1	2	3
2	2	2	3
3	3	3	3

ภาพที่ 3.14 การจัดรูปแบบข้อมูลเมตริกซ์ขนาด 4x4 แบบ Right-angle Block

0	0	1	1	2	2	3	3
0	0	1	1	2	2	3	3
1	1	1	1	2	2	3	3
1	1	1	1	2	2	3	3
2	2	2	2	2	2	3	3
2	2	2	2	2	2	3	3
3	3	3	3	3	3	3	3
3	3	3	3	3	3	3	3

ภาพที่ 3.15 การจัดรูปแบบข้อมูลเมตริกซ์ขนาด 8x8 แบบ Right-angle Block

จากตารางที่ 3.4 ซึ่งแสดงจำนวน Computation และ Communication ของการประมวลผลแบบขนานด้วยการจัดรูปแบบข้อมูลแบบใหม่ที่มีชื่อว่า Right-angle Block เมื่อทำการแก้ปัญหาสมการเชิงเส้นด้วยวิธี Crout Decomposition โดยในตารางมีขนาดของข้อมูลที่แตกต่างซึ่งทำให้เห็นความแตกต่างกันของจำนวน Computation และ Communication ที่เกิดขึ้นเมื่อขนาดของข้อมูล

ตารางที่ 3.4 แสดงจำนวน Computation และ Communication ของการประมวลผลแบบขนานด้วยการจัดรูปแบบข้อมูลแบบ Right-angle Block เมื่อทำการแก้ปัญหาสมการเชิงเส้นด้วยวิธี Crout Decomposition

ขนาดของเมตริกซ์	จำนวน Computation	จำนวน Communication
2x2	3	1
3x3	5	5
4x4	21	14
5x5	36	29
6x6	65	45
7x7	114	56
8x8	183	56

จากตารางที่ 3.4 จะเห็นว่าจำนวนของ Computation และ Communication ที่เกิดขึ้นเมื่อทำการประมวลผลแบบขนานด้วยการจัดรูปแบบข้อมูลแบบ Right-angle Block ซึ่งเป็นวิธีการจัดรูปแบบข้อมูลที่ได้สร้างขึ้นมาใหม่นั้น จะมีจำนวนของ Computation และ Communication น้อยกว่าการประมวลผลแบบขนานด้วยการจัดรูปแบบข้อมูลแบบ Column Block ในตารางที่ 3.2 และแบบ Row Block ในตารางที่ 3.3 (ตัวอย่างวิธีการคำนวณหา Computation และ Communication ดูได้จาก ภาคผนวก ก) ในการหาค่าของผลรวมระหว่างจำนวน Computation กับ Communication ของการจัดรูปแบบข้อมูลแบบ Right-angle Block จะอยู่ในรูป $\frac{n^3}{3} + O(n^2)$

ดังนั้นถ้าหากขนาดของข้อมูลมีขนาดที่ใหญ่ขึ้น ก็จะทำให้เห็นความแตกต่างของเวลาที่ใช้ในการประมวลผลระหว่างการประมวลผลแบบขนานด้วยการจัดรูปแบบข้อมูลแบบ Right-angle Block กับการจัดรูปแบบข้อมูลแบบ Column Block และแบบ Row Block ได้อย่างชัดเจนแม้ว่าค่าของผลรวมระหว่างจำนวน Computation กับ Communication ของการจัดรูปแบบข้อมูลทั้ง 3 แบบ

จะอยู่ในรูป $\frac{n^3}{3} + O(n^2)$ เหมือนกัน แต่ถ้าหากทำการเปรียบเทียบค่าของผลรวมระหว่างจำนวน Computation กับ Communication ในตารางที่ 3.2, 3.3 และ 3.4 จะเห็นถึงความแตกต่างของผลรวมระหว่างจำนวน Computation กับ Communication ที่เกิดขึ้นของทั้ง 3 รูปแบบ

จากการวิเคราะห์ประสิทธิภาพในการทำงานก่อนทำการทดลองจะเห็นว่าการประมวลผลแบบขนานนั้นจะมีประสิทธิภาพการทำงานในเรื่องความเร็วของการประมวลผลข้อมูลดีกว่าการประมวลผลแบบลำดับ และการประมวลแบบขนานด้วยการจัดรูปแบบข้อมูลแบบ Right-angle Block จะมี Speedup ดีกว่าการประมวลผลแบบขนานด้วยการจัดรูปแบบข้อมูลแบบ Column Block และแบบ Row Block ในขณะที่การจัดรูปแบบข้อมูลแบบ Row Block ก็จะมี Speedup ที่ดีกว่า Column Block เช่นกันแม้ว่าสมการที่ใช้ในการหาค่าของ Computation และ Communication อาจจะมีรูปแบบของสมการที่ไม่แตกต่างกันมากนักเพราะสมการที่ได้เป็นสมการที่ใช้ในการหาค่าโดยประมาณที่จะทำให้เกิด Computation และ Communication สูงสุดเท่านั้น

บทที่ 4

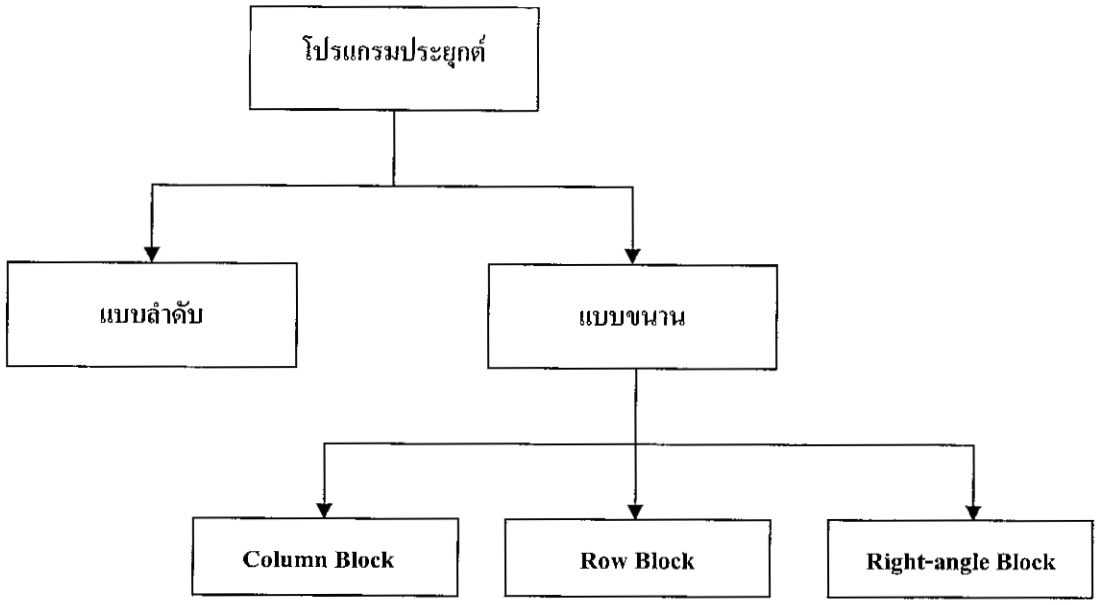
การทดลองและผลการทดลอง

จากกระบวนการที่ได้ทำการวิเคราะห์ และออกแบบไว้แล้วในบทที่ 3 จะนำมาพัฒนาเป็นโปรแกรมประยุกต์ที่ใช้แก้ปัญหาสมการเชิงเส้นด้วยวิธี Crout Decomposition ซึ่งโปรแกรมประยุกต์ที่ได้ทำการพัฒนาขึ้นนี้จะนำมาใช้ทำการทดลองเพื่อหา

1. ความเร็วที่ใช้ในการประมวลผลข้อมูล (Execution Time) ของโปรแกรมประยุกต์ที่ทำการประมวลผลแบบลำดับ กับโปรแกรมประยุกต์ที่ประมวลแบบขนาน
2. อัตราความเร็วที่เพิ่มขึ้น (Speedup) ของโปรแกรมประยุกต์ที่ได้ทำการประมวลผลแบบขนานที่สามารถประมวลผลได้เร็วกว่าโปรแกรมประยุกต์ที่ทำการประมวลผลแบบลำดับเป็นจำนวนกี่เท่า

โดยขั้นตอนในการทดลองจะแบ่งเป็น 4 การทดลองซึ่งจะทดลองกับโปรแกรมประยุกต์ที่ใช้แก้ปัญหาสมการเชิงเส้น Ax=b ด้วยวิธี Crout Decomposition ที่พัฒนาขึ้นเอง (โปรแกรมประยุกต์ที่พัฒนาขึ้นเพื่อใช้ในการทดลองในงานวิจัยนี้สามารถขอได้จากทางคณะวิทยาศาสตร์ มหาวิทยาลัยอุบลราชธานี) โดยจะประกอบด้วย 4 โปรแกรมดังนี้

1. โปรแกรมประยุกต์ที่ทำการประมวลผลแบบลำดับ
2. โปรแกรมประยุกต์ที่ทำการประมวลผลแบบขนานและใช้การจัดรูปแบบข้อมูลแบบ Column Block
3. โปรแกรมประยุกต์ที่ทำการประมวลผลแบบขนานและใช้การจัดรูปแบบข้อมูลแบบ Row Block
4. โปรแกรมประยุกต์ที่ทำการประมวลผลแบบขนานและใช้การจัดรูปแบบข้อมูลแบบ Right-angle Block



ภาพที่ 4.1 ขั้นตอนในการทดลอง

4.1 การทดลอง

โดยในการทดลองได้ทำการทดลองโปรแกรมประยุกต์ที่ทำการประมวลผลผลแบบลำดับบนเครื่องคอมพิวเตอร์ 1 เครื่อง ในขณะที่โปรแกรมประยุกต์ที่ทำการประมวลผลแบบขนานจะถูกประมวลผลบนระบบคลัสเตอร์ขนาด 4 โหนด โดยระบบคลัสเตอร์ที่ได้ใช้เพื่อทำการทดลองในงานวิจัยนี้ตั้งอยู่ที่ห้อง LAB Cluster ชั้น 2 อาคารวิจัย มหาวิทยาลัยอุบลราชธานี ซึ่งโครงสร้างของระบบคลัสเตอร์ประกอบด้วย

1. ส่วนประกอบฮาร์ดแวร์
   - ชุดคอมพิวเตอร์เพนเทียมโฟร์ 2.40 กิกะเฮิร์ต (Pentium 4 2.41 GHz.) 5 เครื่อง
   - การ์ดเน็ทเวิร์ค (LAN Card) จำนวน 6 ใบ ความเร็ว 10/100 Mbps (ที่ Front-end Node มี network card 2 ตัว)
   - เน็ทเวิร์คสวิทช์ (Network Switch) จำนวน 1 ตัว ความเร็ว 10/100 Mbps

โดยมีเครื่องคอมพิวเตอร์ 1 เครื่องทำหน้าที่เป็นโหนดควบคุมเรียกว่า Front-end Node และเครื่องคอมพิวเตอร์ที่เหลืออีก 4 เครื่องทำหน้าที่ประมวลผลเป็นหลัก ซึ่งเรียกว่า Computer Node โดยมีการเชื่อมโยงคอมพิวเตอร์ทั้ง 5 เครื่องเข้าหากันผ่านทาง Network Switch ด้วยสถาปัตยกรรมเครือข่ายแบบ Star

2. ส่วนประกอบของซอฟต์แวร์
   - ระบบปฏิบัติการลีนุกส์ที่ใช้เคอร์เนลตั้งแต่เวอร์ชัน 2.6
   - คอมไพเลอร์ภาษาซี / ซีพลัสพลัสที่ทำงานบนระบบปฏิบัติการลีนุกส์
   - OSCAR 3.0
   - ไลบารี่ MPI

ในส่วนของซอฟต์แวร์จะใช้ Linux Fedora core 1 ทำหน้าที่เป็นระบบปฏิบัติการของเครื่องคอมพิวเตอร์ทั้ง 5 และใช้ OSCAR ทำหน้าที่เป็นตัวควบคุมการทำงานของเครื่องคอมพิวเตอร์ทั้ง 5 บนระบบคลัสเตอร์

ส่วนข้อมูลที่จะใช้ในการทดลองในงานวิจัยนี้ เป็นข้อมูลที่ได้จากการสุ่มตัวเลขด้วยโปรแกรมประยุกต์ที่ได้พัฒนาขึ้น ซึ่งจะทำการสุ่มข้อมูลออกมาให้อยู่ในรูปของเมตริกซ์ A และเวคเตอร์ b ขนาดต่าง ๆ ตามที่ได้กำหนดไว้ในตารางที่ 4.1 ซึ่งจะเป็นขนาดของข้อมูลที่จะใช้ในการทดลองในงานวิจัยนี้

ตารางที่ 4.1 แสดงขนาดข้อมูลของเมตริกซ์ A และ เวคเตอร์ b

ขนาดของเมตริกซ์ A	ขนาดข้อมูลของเวคเตอร์ b
4x4	4x1
8x8	8x1
16x16	16x1
32x32	32x1
64x64	64x1
128x128	128x1
256x256	256x1
512x512	512x1
1024x1024	1024x1
2048x2048	2048x1
4096x4096	4096x1
8192x8192	8192x1

โดยในการวัดประสิทธิภาพการประมวลผลของโปรแกรมประยุกต์แต่ละแบบนั้นจะวัดจาก

1. Execution Time คือเวลาที่ใช้ในการประมวลผลข้อมูล

2. Speedup คืออัตราเร็วเพิ่มขึ้นของความเร็วซึ่งเป็นค่าที่บ่งบอกว่าโปรแกรมประยุกต์ที่ประมวลผลแบบขนานจะสามารถประมวลผลได้เร็วกว่าโปรแกรมประยุกต์ที่ประมวลผลแบบลำดับเป็นจำนวนกี่เท่า [25, 26] โดย Speedup หาได้จาก

$$S = \frac{Ts}{Tp} \tag{4.1}$$

เมื่อ S = Speedup

Ts = Sequential Runtime คือเวลาที่โปรแกรมประยุกต์ที่ประมวลผลแบบลำดับใช้ประมวลผล

Tp = Parallel Runtime คือเวลาที่โปรแกรมประยุกต์ที่ประมวลผลแบบขนานใช้ประมวลผล

วิธีการทดลอง

1. นำข้อมูลที่ได้จากการสุ่มตัวเลขตามขนาดที่กำหนดไว้ในตารางที่ 4.1 จะนำไปใช้เป็น Input ของโปรแกรมประยุกต์ที่ได้พัฒนาขึ้นทั้ง 4 โปรแกรม

2. จดบันทึกเวลาที่ใช้ในการประมวลผล

3. หา Speedup ของโปรแกรมประยุกต์ที่ประมวลผลแบบขนานจากสมการที่ 4.1

4. สร้างกราฟเวลาที่ใช้ในการประมวลผลและSpeedup ของโปรแกรมประยุกต์ทั้ง 4 โปรแกรม

5. เปรียบเทียบความเร็วในการประมวลผลข้อมูลระหว่างโปรแกรมประยุกต์ที่ประมวลผลแบบลำดับ กับโปรแกรมประยุกต์ที่ประมวลผลแบบขนาน

6. เปรียบเทียบ Speedup ของโปรแกรมประยุกต์ที่ประมวลแบบขนานแต่ละแบบ

หมายเหตุ ในการทดลองจะต้องมีการนำผลการทดลองที่ได้ไปทำการสร้างเป็นกราฟซึ่งมีความจำเป็นจะต้องใช้ชื่อย่อ ดังนั้นจึงต้องมีการกำหนดความหมายของชื่อย่อที่จะใช้ดังนี้

1. Sequential คือโปรแกรมประยุกต์ที่ประมวลผลแบบลำดับ

2. Column คือโปรแกรมประยุกต์ที่ประมวลผลแบบขนานด้วยวิธีการจัดรูปแบบข้อมูลแบบ Column Block

3. Row คือโปรแกรมประยุกต์ที่ประมวลผลแบบขนานด้วยวิธีการจัดรูปแบบข้อมูลแบบ Row Block

4. Right-angle คือโปรแกรมประยุกต์ที่ประมวลผลแบบขนานด้วยวิธีการจัดรูปแบบข้อมูลแบบ Right-angle Block

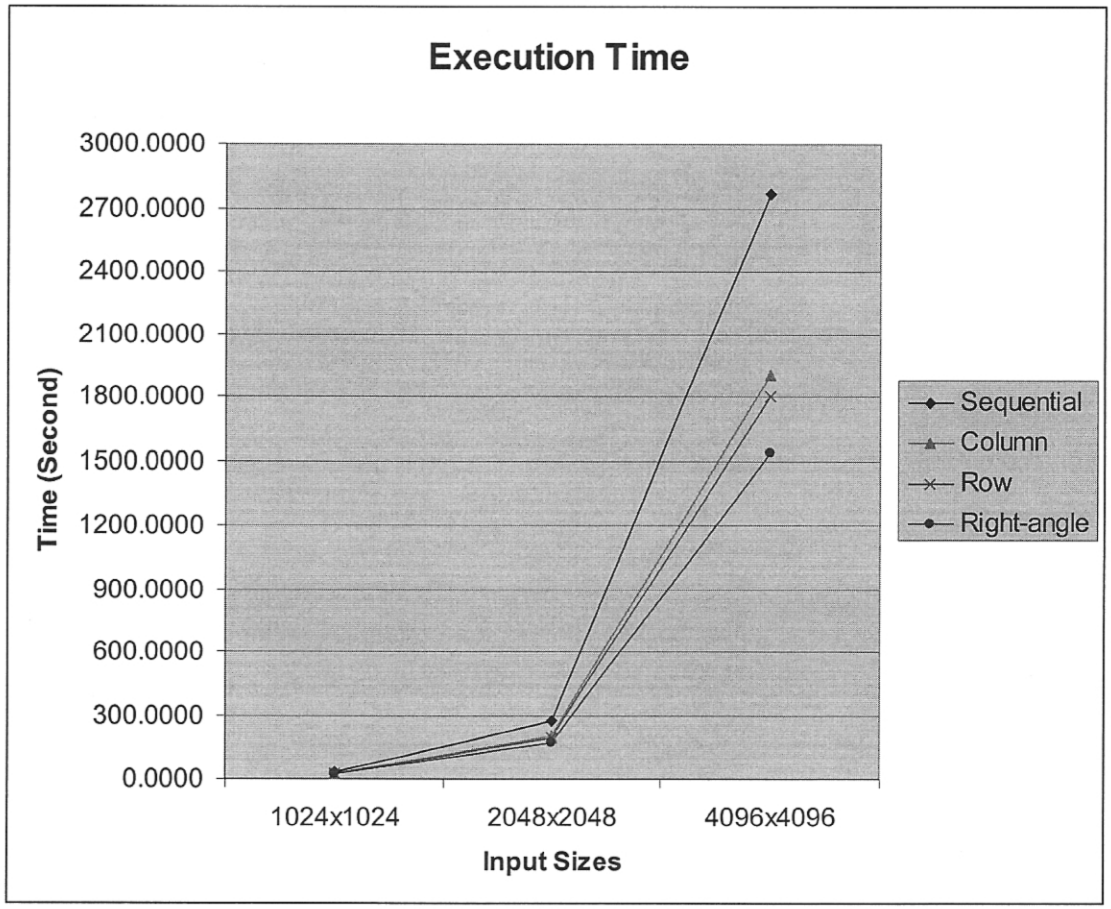
4.2 ผลการทดลอง

4.2.1 เวลาที่ใช้ในการประมวลผลข้อมูล

ผลที่ได้จากการทดลองเพื่อวัดเวลาที่ใช้ในการประมวลผลข้อมูลของโปรแกรมประยุกต์ที่ประมวลผลแบบลำดับ และโปรแกรมประยุกต์ที่ประมวลผลแบบขนานได้แสดงไว้ในตารางที่ 4.2

ตารางที่ 4.2 เวลาที่ใช้ในการประมวลผลข้อมูลของโปรแกรมประยุกต์ที่ประมวลผลแบบลำดับ กับโปรแกรมประยุกต์ที่ประมวลผลข้อมูลขนานด้วยการจัดรูปแบบข้อมูลทั้ง 3 รูปแบบ

ขนาดของเมตริกซ์	เวลาที่ใช้ในการประมวลผล (วินาที)			
	การประมวลผลแบบลำดับ	การประมวลผลแบบขนาน		
	Sequential	**Column Block**	**Row Block**	**Right-angle Block**
4x4	0.0000	0.0005	0.0005	0.0004
8x8	0.0000	0.0011	0.0009	0.0007
16x16	0.0000	0.0029	0.0026	0.0019
32x32	0.0001	0.0083	0.0080	0.0041
64x64	0.0010	0.0212	0.0206	0.0129
128x128	0.0081	0.0743	0.0734	0.0704
256x256	0.0674	0.2988	0.2832	0.2651
512x512	1.8496	2.3572	2.2985	2.1323
1024x1024	30.4834	24.4895	24.0994	20.5926
2048x2048	271.6799	199.4923	195.6947	168.5045
4096x4096	2769.7218	1903.2906	1801.5853	1535.8385
8192x8192	19049.8748	13417.2991	13261.8719	11700.5911



ภาพที่ 4.2 กราฟแสดงการเปรียบเทียบเวลาที่ใช้ประมวลผลระหว่างการประมวลผลแบบลำดับกับการประมวลผลแบบขนานด้วยการจัดรูปแบบข้อมูลทั้ง 3 รูปแบบ

4.3 เปรียบเทียบประสิทธิภาพการทำงาน

ผลการทดลองจากตารางที่ 4.2 และกราฟในภาพที่ 4.2 จะนำมาใช้เพื่อเป็นข้อมูลสำหรับการเปรียบเทียบการทำงานของโปรแกรมประยุกต์ที่ประมวลผลแบบลำดับและโปรแกรมประยุกต์ที่ประมวลผลแบบขนาน ซึ่งโปรแกรมประยุกต์ที่ทำการประมวลผลแบบขนานก็จะแบ่งเป็น 3 รูปแบบด้วยกัน การเปรียบเทียบประสิทธิภาพการทำงานในงานวิจัยชิ้นนี้จะใช้ความเร็วในการประมวลผลและ Speedup เป็นตัวบ่งชี้ถึงประสิทธิภาพการทำงานของโปรแกรมประยุกต์

4.3.1 ความเร็วในการประมวลผล

ความเร็วในการประมวลผล คือเวลาจริงที่โปรแกรมใช้ในการประมวลผลข้อมูลตั้งแต่การเริ่มต้นประมวลผลจนกระทั่งสิ้นสุดการประมวลผลซึ่งถ้าหากโปรแกรมประยุกต์รูปแบบใดใช้เวลาในการประมวลผลน้อยก็แสดงว่าโปรแกรมประยุกต์รูปแบบนั้นมีความเร็วในการประมวลผลที่ดีเช่นกัน

4.3.1.1 การเปรียบเทียบเวลาที่ใช้ในการประมวลผลข้อมูลระหว่างการประมวลผลแบบลำดับ กับการประมวลผลแบบขนาน

การเปรียบเทียบการประมวลผลข้อมูลระหว่างการประมวลผลแบบลำดับกับการประมวลผลแบบขนานนั้นจะนำเอาเวลาที่ใช้ในการประมวลผลข้อมูลของโปรแกรมประยุกต์ที่ประมวลผลข้อมูลแบบลำดับ และโปรแกรมประยุกต์ที่ประมวลผลข้อมูลแบบขนานแบบต่าง ๆ ในตารางที่ 4.2 มาใช้ทำการเปรียบเทียบการประมวลผลข้อมูล นอกจากนี้ยังได้ใช้กราฟเพื่อช่วยทำให้มองเห็นภาพได้ชัดเจนยิ่งขึ้นโดยสามารถดูได้จากภาพที่ 4.2

จากตารางที่ 4.2 และภาพที่ 4.2 แสดงให้เห็นว่า

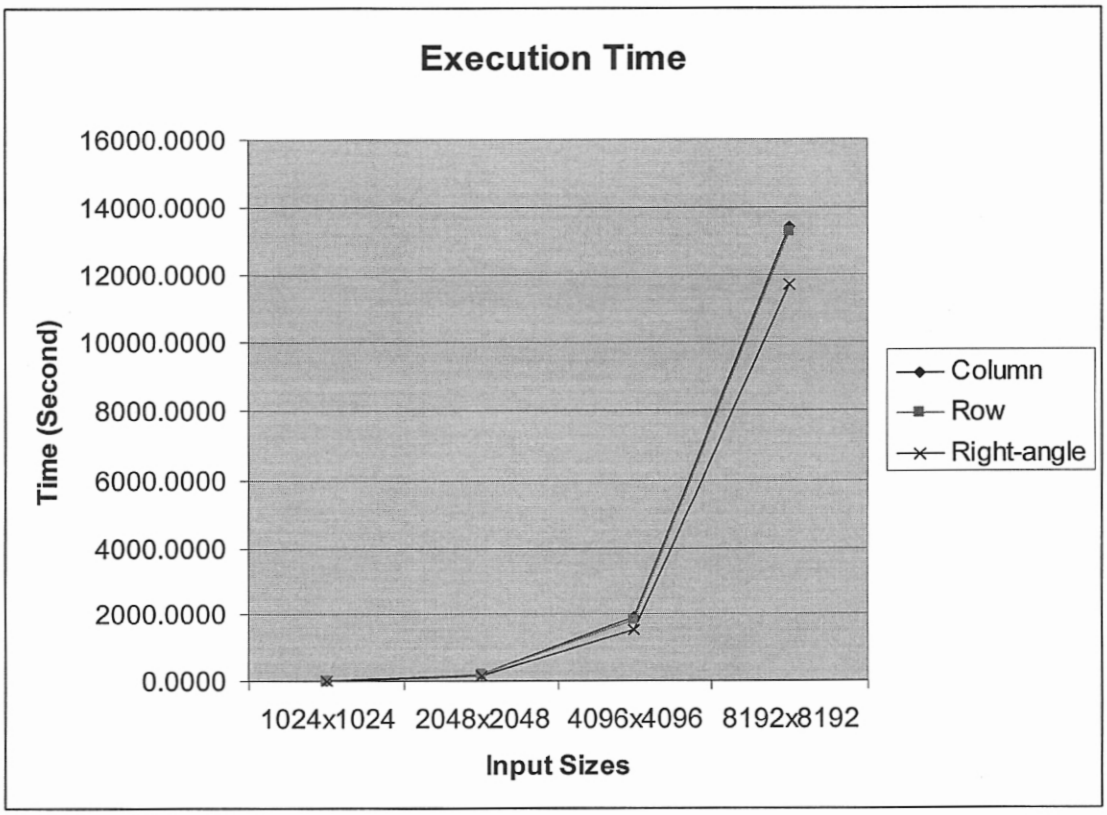
1. ถ้าข้อมูลที่ใช้เป็น Input Size มีขนาดเล็ก คืออยู่ในช่วง 4 < Input Size < 1024 โปรแกรมประยุกต์ที่ประมวลผลแบบลำดับจะใช้เวลาในการประมวลผลน้อยกว่าโปรแกรมประยุกต์ที่ประมวลผลแบบขนานทุกรูปแบบ

2. ถ้าข้อมูลที่ใช้เป็น Input Size มีขนาดใหญ่กว่าหรือเท่ากับ 1024 โปรแกรมประยุกต์ที่ประมวลผลแบบลำดับจะใช้เวลาในการประมวลผลมากกว่าโปรแกรมประยุกต์ที่ประมวลผลแบบขนานทุกรูปแบบ และยิ่งข้อมูลมีขนาดใหญ่ขึ้น โปรแกรมประยุกต์ที่ประมวลผลแบบลำดับก็จะใช้เวลาในการประมวลผลมากกว่าโปรแกรมประยุกต์ที่ประมวลผลแบบขนานมากขึ้นเรื่อย ๆ

ดังนั้นจากผลการทดลองอาจกล่าวได้ว่าโปรแกรมประยุกต์ที่ประมวลผลแบบลำดับจะใช้เวลาในการประมวลผลน้อยกว่าโปรแกรมประยุกต์ที่ประมวลผลแบบขนานแต่ถ้าข้อมูลมีขนาดใหญ่ขึ้นจนถึงระดับหนึ่งโปรแกรมประยุกต์ที่ประมวลผลแบบขนานจะใช้เวลาในการประมวลผลน้อยกว่าโปรแกรมประยุกต์ที่ประมวลผลแบบลำดับเพราะการประมวลผลแบบลำดับไม่มี Communication เกิดขึ้นในขณะที่มีการประมวลผลทำให้ช่วยลดเวลาที่ใช้ในการประมวลผลได้ทำให้หลายงานวิจัยที่เกี่ยวข้องกับการประมวลผลแบบขนาน ได้กล่าวเอาไว้ว่า “การประมวลผลแบบลำดับ จะใช้เวลาในการประมวลผลข้อมูลน้อยกว่าการประมวลผลแบบขนานเมื่อขนาดของข้อมูลมีขนาดเล็กแต่ถ้าขนาดของข้อมูลเพิ่มขึ้นจนกระทั่งถึงระดับหนึ่งการประมวลผลแบบลำดับจะใช้เวลาในการประมวลผลข้อมูลมากกว่าการประมวลผลแบบขนาน” [23, 24] จากผลการทดลองในตารางที่ 4.2 ในบทที่ 4 ของงานวิจัยนี้ก็สอดคล้องกับคำพูดดังกล่าว

4.3.1.2 การเปรียบเทียบเวลาที่ใช้ในการประมวลผลระหว่างการจัดรูปแบบข้อมูลแบบ Column Block, Row Block และRight-angle Block

จากตารางที่ 4.2 ทำให้ทราบว่าวิธีการจัดรูปแบบข้อมูลที่แตกต่างกันนั้นมีผลต่อเวลาที่ใช้ในการประมวลผลข้อมูลโดยการจัดรูปแบบข้อมูลแบบ Right-angle Block จะใช้เวลาในการประมวลผลน้อยกว่าการจัดรูปแบบข้อมูลแบบ Column Block และการจัดรูปแบบข้อมูลแบบ Row Block ไม่ว่าข้อมูลจะมีขนาดเท่าใด นอกจากนี้การจัดรูปแบบข้อมูลแบบ Row Block ก็ใช้เวลาในการประมวลผลข้อมูลน้อยกว่าการจัดรูปแบบข้อมูลแบบ Column Block ซึ่งเมื่อนำข้อมูลจากตารางที่ 4.2 มาสร้างกราฟเพื่อให้เห็นความแตกต่างของเวลาที่ใช้ในการประมวลผลแบบขนานด้วยการจัดรูปแบบข้อมูลที่แตกต่างกันจึงได้กราฟดังภาพที่ 4.3



ภาพที่ 4.3 กราฟแสดงการเปรียบเทียบเวลาที่ใช้ในประมวลผลระหว่างการประมวลผลแบบขนานด้วยการจัดรูปแบบข้อมูลแบบ Column Block, Row Block และRight-angle Block

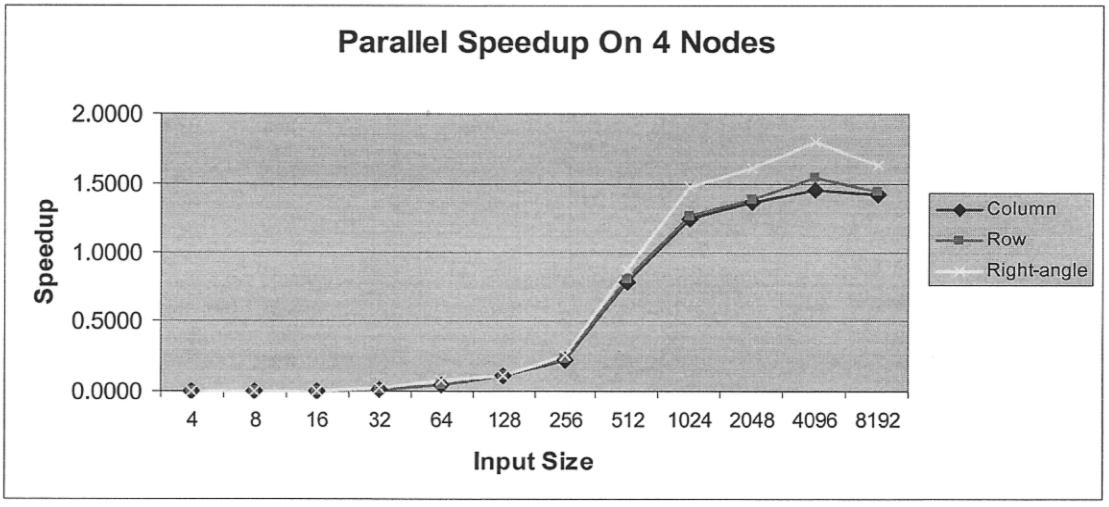
4.3.2 Speedup

Speedup คืออัตราเร็วเพิ่มขึ้นของความเร็วซึ่งเป็นค่าที่จะบ่งบอกว่าโปรแกรมประยุกต์ที่ประมวลผลแบบขนานจะสามารถประมวลผลได้เร็วกว่าโปรแกรมประยุกต์ที่ประมวลผลแบบลำดับเป็นจำนวนกี่เท่า [25, 26] โดย Speedup หาได้จากสมการที่ 4.1

การหา Speedup ในงานวิจัยนี้ได้นำผลการทดลองจากตารางที่ 4.2 ซึ่งเป็นเวลาที่ใช้ในการประมวลผลข้อมูลของโปรแกรมประยุกต์ที่ประมวลผลแบบขนานบนระบบคลัสเตอร์ที่ประกอบด้วยโหนดจำนวนทั้งหมด 4 โหนด ซึ่งจากการนำค่าในตารางที่ 4.2 มาใช้เพื่อหาค่าของ Speedup ของการประมวลผลแบบขนานด้วยสมการที่ 4.1 ทำให้ได้ค่าของ Speedup ดังที่ได้แสดงไว้ในตารางที่ 4.3 และเมื่อนำข้อมูลจากตารางที่ 4.3 ไปสร้างกราฟเพื่อให้เห็นแนวโน้มของ Speedup ของแต่ละแบบ ก็จะทำให้ได้กราฟดังภาพที่ 4.4

ตารางที่ 4.3 แสดงค่า Speedup ของโปรแกรมประยุกต์ที่ประมวลผลแบบขนานด้วยการจัดรูปแบบข้อมูลทั้ง 3 รูปแบบ

ขนาดของเมตริกซ์	Parallel Speedup on 4 Nodes		
	Column Block	Row Block	Right-angle Block
4x4	0.0000	0.0000	0.0000
8x8	0.0000	0.0000	0.0000
16x16	0.0000	0.0000	0.0000
32x32	0.0120	0.0125	0.0244
64x64	0.0472	0.0485	0.0775
128x128	0.1090	0.1104	0.1151
256x256	0.2256	0.2380	0.2542
512x512	0.7847	0.8047	0.8674
1024x1024	1.2448	1.2649	1.4803
2048x2048	1.3619	1.3883	1.6123
4096x4096	**1.4552**	**1.5374**	**1.8034**
8192x8192	1.4198	1.4364	1.6281



ภาพที่ 4.4 กราฟแสดงการเปรียบเทียบ Speedup ของโปรแกรมประยุกต์ที่ประมวลผลแบบขนานทั้ง 3 รูปแบบ

จากตารางที่ 4.3 และกราฟในภาพที่ 4.4 จะสังเกตว่าโปรแกรมประยุกต์ที่ประมวลผลแบบขนานด้วยการจัดรูปแบบข้อมูลแบบ Right-angle Block นั้นสามารถจะทำ Speedup ได้สูงกว่าการจัดรูปแบบข้อมูลแบบ Column Block และแบบ Row Block ทุกขนาดข้อมูล โดยที่ Speedup ของการจัดรูปแบบข้อมูลแบบ Right-angle Block จะให้ค่า Speedup ที่สูงกว่าการจัดรูปแบบข้อมูลแบบ Column Block และแบบ Row Block เล็กน้อยเมื่อขนาดของข้อมูลมีขนาดเล็กซึ่งจากกราฟในภาพที่ 4.4 แสดงให้เห็นว่าเมื่อขนาดของข้อมูลมีขนาดใหญ่กว่า 256 ค่าของ Speedup ก็จะเริ่มสูงขึ้นเรื่อย ๆ จนกระทั่งขนาดข้อมูลมีขนาดใหญ่ถึงระดับหนึ่ง Speedup ก็จะให้ค่าสูงสุด แม้ว่าขนาดของข้อมูลจะใหญ่ขึ้นอีก Speedup ก็จะไม่สูงไปกว่านี้อีกแล้วโดยเมื่อข้อมูลมีขนาด 4096 ก็จะเป็นขนาดของข้อมูลที่ Speedup ทำได้สูงสุด ซึ่งเมื่อขนาดข้อมูลใหญ่กว่า 4096 Speedup ก็ไม่ได้สูงขึ้นไปอีกในขณะที่การจัดรูปแบบข้อมูลแบบ Row Block ก็จะมี Speedup สูงกว่าการจัดรูปแบบข้อมูลแบบ Column Block แต่สูงกว่าไม่มากเท่าไหร่นัก

หากจะเปรียบเทียบ Speedup ของโปรแกรมประยุกต์ที่ประมวลผลแบบขนานด้วยการจัดรูปแบบข้อมูลทั้ง 3 รูปแบบ จะสังเกตเห็นว่าโปรแกรมประยุกต์ที่ประมวลผลแบบขนานด้วยการจัดรูปแบบข้อมูลแบบ Right-angle Block สามารถที่จะให้ค่าของ Speedup ได้สูงที่สุดในบรรดาการจัดรูปแบบข้อมูลทั้ง 3 รูปแบบโดยจะมี Speedup สูงสุดอยู่ที่ประมาณ 1.8 เท่าเมื่อขนาดของข้อมูลเท่ากับ 4096 ในขณะที่การจัดรูปแบบข้อมูลแบบ Row Block สามารถจะให้ค่าของ Speedup ได้เป็น

อันดับที่ 2 และการจัดรูปแบบข้อมูลแบบ Column Block นั้นสามารถที่จะให้ค่าของ Speedup ได้เป็นอันดับที่ 3 ตามลำดับ

ดังนั้น โปรแกรมประยุกต์ที่ประมวลผลแบบขนานด้วยการจัดรูปแบบข้อมูลแบบ Right-angle Block จะให้ค่าของ Speedup สูงกว่าโปรแกรมประยุกต์ที่ประมวลผลแบบขนานด้วยการจัดรูปแบบข้อมูลแบบ Column Block และแบบ Row Block

บทที่ 5
สรุปผลการศึกษา

5.1 สรุปผลการศึกษา

งานวิจัยนี้ต้องการศึกษาวิธีการประมวลผลแบบขนานบนระบบคลัสเตอร์ ซึ่งเทคนิคการจัดรูปแบบข้อมูล คือพื้นฐานสำคัญ ดังนั้นการศึกษาการจัดรูปแบบข้อมูลแบบ Column Block และแบบRow Block จะทำให้ทราบถึงขั้นตอนการทำงานตลอดจนเทคนิควิธีการจัดรูปแบบข้อมูล และข้อบกพร่องของการจัดรูปแบบข้อมูล ซึ่งจะนำไปสู่การปรับปรุงข้อบกพร่องของเทคนิคการจัดรูปแบบข้อมูล เพื่อจะทำให้ได้เทคนิคการจัดรูปแบบข้อมูลแบบใหม่ เพื่อที่จะนำมาประยุกต์ใช้กับการแก้ปัญหาสมการเชิงเส้นโดยวิธี Crout Decomposition ให้มีประสิทธิภาพในการทำงานมากกว่าการจัดรูปแบบข้อมูลแบบ Column Block และ Row Block ซึ่งจะใช้เวลาที่ใช้ในการประมวลผล และ Speedup เป็นสิ่งที่จะใช้ชี้วัดประสิทธิภาพในงานวิจัยนี้ โดยในการทดลองการประมวลผลแบบขนานบนระบบคลัสเตอร์ในงานวิจัยนี้ ได้พัฒนาโปรแกรมประยุกต์ที่ประมวลผลแบบขนาน เพื่อใช้แก้ปัญหาสมการเชิงเส้น Ax=b โดยโปรแกรมประยุกต์ที่พัฒนาขึ้นจะใช้วิธี Crout Decomposition ในการแก้ปัญหาสมการเชิงเส้นนี้ และในการทดลองในงานวิจัยนี้ได้ทำการทดสอบการประมวลผลแบบขนานบนระบบคลัสเตอร์ขนาด 4 โหนด และใช้ข้อมูลที่อยู่ในรูปของเมตริกซ์ที่มีขนาด 4x4, 8x8, 16x16, 32x32, 64x64, 128x128, 256x256, 512x512, 1024x1024, 2048x2048, 4096x409 และ 8192x8192 ในการทดลองในงานวิจัยนี้

จากการทดลองการวัดประสิทธิภาพความเร็วในการประมวลผล ในตารางที่ 4.2 และกราฟในภาพที่ 4.2 พบว่าโปรแกรมประยุกต์ที่ประมวลผลแบบขนานมีประสิทธิภาพความเร็วในการประมวลผลข้อมูลดีกว่า โปรแกรมประยุกต์ที่ประมวลผลแบบลำดับ นอกจากนี้พบว่าโปรแกรมประยุกต์ที่ใช้การจัดรูปแบบข้อมูลแบบ Right-angle Block ซึ่งเป็นการจัดรูปแบบข้อมูลแบบใหม่ที่ได้พัฒนาขึ้น ก็ให้ประสิทธิภาพความเร็วในการประมวลผลดีกว่าโปรแกรมประยุกต์ที่ใช้การจัดรูปแบบข้อมูลแบบ Column Block และการจัดรูปแบบข้อมูลแบบ Row Block และจากการทดลองการวัดประสิทธิภาพ Speedup ของการประมวลผลแบบขนานด้วยการจัดรูปแบบข้อมูลที่แตกต่างกันโดยจากตารางที่ 4.3 และจากกราฟในภาพที่ 4.4 พบว่าโปรแกรมประยุกต์ที่ใช้การจัดรูปแบบข้อมูลแบบ Right-angle Block สามารถจะให้ค่า Speedup ที่สูงกว่าการจัดรูปแบบข้อมูล

แบบ Column Block และแบบ Row Block เมื่อข้อมูลมีขนาดใหญ่กว่า 16x16 และจะให้ค่า Speedup สูงสุดเมื่อข้อมูลมีขนาด 4096x4096

สรุปว่าการจัดรูปแบบข้อมูลแบบ Right-angle Block นั้นจะให้ประสิทธิภาพการทำงานที่ดีกว่าการจัดรูปแบบข้อมูลแบบ Column Block และการจัดรูปแบบข้อมูลแบบ Row Block ในขณะที่การจัดรูปแบบข้อมูลแบบ Row Block ซึ่งจากการทดลองในงานวิจัยนี้ได้ทำให้ทราบขั้นตอนในการพัฒนาโปรแกรมประยุกต์ที่ประมวลผลแบบขนานบนระบบคลัสเตอร์ และเข้าใจถึงการจัดรูปแบบข้อมูลซึ่งเป็นพื้นฐานที่สำคัญ สำหรับการพัฒนาโปรแกรมประยุกต์แบบขนาน เพื่อจะทำให้โปรแกรมประยุกต์ที่ประมวลผลแบบขนานทำงานได้อย่างมีประสิทธิภาพ นอกจากนี้ยังทำให้เห็นตัวอย่างการนำวิธีการประมวลผลแบบขนานบนระบบคลัสเตอร์ไปประยุกต์ใช้ให้เกิดประโยชน์ โดยความรู้ความเข้าใจที่ได้จากงานวิจัยนี้ จะนำไปพัฒนาให้เป็นโปรแกรมประยุกต์สำเร็จรูปแบบขนานที่ง่ายต่อการใช้งาน เพื่อช่วยแก้ปัญหาทางด้านคณิตศาสตร์, วิทยาศาสตร์ หรือ ศาสตร์ทางด้านสาขาอื่น ๆ ที่สามารถนำวิธีการแก้ปัญหาสมการเชิงเส้นไปช่วยแก้ปัญหานั้น ๆ ได้ เช่น การหาค่าแรงดันและกระแสไฟฟ้าในงานวิศวกรรมไฟฟ้า หรือการนำไปพัฒนาเป็นโปรแกรมประยุกต์สำหรับใช้เพื่อพยากรณ์ข้อมูลทางธุรกิจ เช่น การวิเคราะห์และการตัดสินใจในการซื้อขายหลักทรัพย์ในตลาดหุ้น

5.2 ข้อเสนอแนะ

5.2.1 ในงานวิจัยนี้ได้ทำการออกแบบวิธีการจัดรูปแบบข้อมูลแบบใหม่ขึ้นมา 1 รูปแบบ และได้ทำการทดสอบประสิทธิภาพการทำงานซึ่งจากการทดสอบพบว่าการจัดรูปแบบข้อมูลแบบใหม่ให้ประสิทธิภาพในการทำงานดีกว่าการจัดรูปแบบข้อมูลแบบ Column Block และแบบ Row Block ดังนั้นในงานวิจัยชิ้นต่อไปควรจะมีการออกแบบวิธีการจัดรูปแบบข้อมูลรูปแบบอื่น ๆ ที่ให้ประสิทธิภาพในการทำงานดีกว่าการจัดรูปแบบข้อมูลในงานวิจัยนี้

5.2.2 เนื่องจากสูตรที่ใช้เพื่อทำการคำนวณหาค่าของ Upper Bound ของ Computation และ Communication ในงานวิจัยนี้ยังไม่มีความแม่นยำในการคำนวณหาค่าเหล่านั้น ดังนั้นจึงควรที่จะมีการปรับปรุงสูตรให้มีความถูกต้องแม่นยำมากขึ้น

5.2.3 เนื่องจากในงานวิจัยนี้ได้ทำการทดสอบประสิทธิภาพการจัดรูปแบบข้อมูลบนระบบคลัสเตอร์ขนาด 4 โหนดเท่านั้น ดังนั้นจึงควรที่จะมีการทดสอบประสิทธิภาพการจัดรูปแบบข้อมูลบนระบบคลัสเตอร์ที่มีขนาดใหญ่กว่านี้เพื่อที่จะทำให้เห็นความแตกต่างในการทำงาน

5.2.4 เนื่องจากในงานวิจัยนี้ไม่ได้ทำการทดสอบเพื่อหาค่า Efficiency ที่เกิดขึ้นในขณะที่มีการประมวลผลบนระบบคลัสเตอร์ด้วยจำนวนโหนดที่มีขนาดต่าง ๆ กัน ดังนั้นจึงควรที่จะมีการ

ทดสอบเพื่อหาค่า Efficiency ที่เกิดขึ้นเมื่อมีการประมวลผลบนระบบคลัสเตอร์ด้วยจำนวนโหนดที่มีขนาดต่าง ๆ กันซึ่งจะช่วยให้ทราบว่าระบบคลัสเตอร์ขนาดเท่าใดที่จะให้ประสิทธิภาพในการทำงานสูงสุดกับรูปแบบวิธีการจัดรูปแบบข้อมูลแบบต่าง ๆ

5.2.5 ในงานวิจัยนี้ได้มุ่งที่จะทำการทดสอบการจัดรูปแบบข้อมูลที่ใช้รูปแบบการส่งข้อมูลครั้งละ 1 ข้อมูลไปยังโหนดแต่ละโหนดหากมีการติดต่อกันเกิดขึ้น ดังนั้นควรจะมีการวิจัยถึงประสิทธิภาพการทำงานเมื่อการจัดรูปแบบข้อมูลใช้รูปแบบการส่งข้อมูลมากกว่าครั้งละ 1 ข้อมูล

5.2.6 ในงานวิจัยนี้ได้ใช้วิธี Crout Decomposition ในการแก้ปัญหาสมการเชิงเส้น ซึ่งวิธี Crout Decomposition จะไม่สามารถแก้ปัญหาสมการเชิงเส้นได้ ถ้าในกรณีที่ตัวหารเป็นศูนย์เกิดขึ้น ดังนั้นควรจะมีการศึกษาถึงวิธีการแก้ปัญหากรณีที่ตัวหารเป็นศูนย์ถ้าใช้วิธี Crout Decomposition ในการแก้ปัญหาสมการเชิงเส้น

เอกสารอ้างอิง

เอกสารอ้างอิง

[1] ภุชงค์ อุทโยภาศ. "สร้างซุปเปอร์คอมพิวเตอร์ราคาถูกด้วยเทคโนโลยี คลัสเตอร์," Article. http://hpcnc.cpe.ku.ac.th/article/clusterall.pdf. 2547.

[2] George Em Karniadakis and Robert M. Kirby. Parallel Scientific Computing in C++ and MPI. Cambridg University Press, 2003.

[3] James W. Demmel. "Dense Linear Algebra - I," Applications of Parallel Computers. http://ceprofs.tamu.edu/jzhang/cven-302-05s/chap20.ppt. 1999.

[4] James W. Demmel, Michael T. Heath and Henk A. van der Vorst. Parallel numerical linear algebra. University of California at Berkeley, 1993.

[5] ดำรงค์ ทิพย์โยธา. ตัวกำหนดและเมทริกซ์ : ระเบียบวิธีคณนา. พิมพ์ครั้งที่ 1. กรุงเทพมหานคร : สำนักพิมพ์แห่งจุฬาลงกรณ์มหาวิทยาลัย, 2540.

[6] ดำรงค์ ทิพย์โยธา และเพ็ญพรรณ ยังคง. พีชคณิตเชิงเส้น. พิมพ์ครั้งที่ 5. กรุงเทพมหานคร : สำนักพิมพ์แห่งจุฬาลงกรณ์มหาวิทยาลัย, 2546.

[7] "พีชคณิตเชิงเส้นและสมการพีชคณิตเชิงเส้น," คณิตศาสตร์วิศวกรรม. http://www.geocities.com/sno_math/linear1.pdf. 2548.

[8] มนัส สังวรศิลป์ และวรรัตน์ ภัทรอมรกุล. คู่มือโปรแกรม MATLAB ฉบับสมบูรณ์. พิมพ์ครั้งที่ 1. กรุงเทพมหานคร : ศูนย์พิมพ์พลชัย, 2543.

[9] "Crout's Method," Mathworld. http://mathworld.wolfram.com/CroutsMethod.html. 2005.

[10] ธรา อั่งสกุล, จุลวดี มณีศิลป์ และภุชงค์ อุทโยภาศ. "เทคโนโลยีลีนุกซ์คลัสเตอร์ และระบบเซิร์ฟเวอร์ขนาดใหญ่," Article. http://hpcnc.cpe.ku.ac.th/article/xcluster.pdf. 2547.

[11] "Faster, Cheaper and Better Crash Simulations Improve Vehicle Safety," Solution Summary. http://cache-www.intel.com/cd/00/00/02/95/29545_audi.pdf. 2005.

[12] S. Bae and S. H. Ko. Parallel Wolff Cluster Algorithms. Syracuse University, 1994.

[13] "Protein Modeling," Chemictical Simmulation Group. http://www.chemicalsimulations.com/public/capabilities/prot_model.html. 2005.

[14] "B-dna curvature," DNA representations. http://honiglab.cpmc.columbia.edu/grasp/pictures.html. 2005.

[15] Marc Snir and et al. MPI: The Complete Reference Volume 1 The MPI Core. Second edition. London : The MIT Press, 1998.

[16] บงกช สุขอนันต์. "การหาคำตอบของสมการพีชคณิตเชิงเส้น," Numerical Methods of Engineering. http://gear.eng.ubu.ac.th/%7Ebongkoj/NumericalMethod/ NM_Chapter5.pdf. 2547.

[17] สมัย เหล่าวานิชย์. คณิตศาสตร์ 2 (พื้นฐาน+เพิ่มเติม). กรุงเทพมหานคร : ธีรพงษ์การพิมพ์, 2537.

[18] Jun ZHANG. "Egenvalue," Computer Applications in Engineering and Construction. http://ceprofs.tamu.edu/jzhang/cven-302-05s/chap20.ppt. 2005.

[19] Jirada Kuntraruk. DATA LAYOUT ISSUES FOR EFFICIENT PARALLEL GAUSSIAN ELIMINATION. Master's Thesis : Lehigh University, 1999.

[20] Michael Wasilewski. Parallel Gaussian Elimination. Master's Thesis : Waterloo University, 2004.

[21] Kaya, D and Wright, K. Parallel Algorithms for LU Decomposition on a Shared Memory Multiprocessor. Master's Thesis : University of Newcastle, 1993.

[22] Graham F. Carey. PARALLEL SUPERCOMPUTING:METHODS, ALGORITHMS and APPLICATIONS. New York : British Library Cataloguing in Publication Data available, 1989.

[23] Jens Simon and Jens-Michael Wierum. Sequential Performance versus Scalability: Optimizing Parallel LU-Decomposition. Paderborn University, 1996.

[24] Mohammad Bankazemi and Dhabaleswar K.Panda. Efficient Scatter Communication in wormhole k-ary n-cubes with Multidestination Message Passing. The Ohio State University, 1998.

[25] A. Karp and H. Flatt. Measuring Parallel Processor Performance. Communications of the ACM, 1990.

[26] D. Eager, J. Zahorian and E. Lazowska. Speedup Versus Efficiency in Parallel Systems. IEEE Transactions on Computer, 1989.

ภาคผนวก

ภาคผนวก ก

วิธีการหา Computation และ Communication ของการจัดข้อมูลแบบ Column Block

ภาคผนวก ก

วิธีการหา Computation และ Communication ของการจัดข้อมูลแบบ Column Block

ในส่วนของภาคผนวก ก นี้จะเป็นการแสดงวิธีการหา Computation และ Communication ของการประมวลผลแบบขนานโดยจะขอยกตัวอย่างวิธีการหา Computation และCommunication ของการจัดข้อมูลแบบ Column Block ด้วยวิธีของ Crout Decomposition บนระบบคลัสเตอร์ที่ประกอบด้วย 4 โหนด และใช้เมตริกซ์ขนาด 4x4 เป็นขนาดข้อมูลตัวอย่าง (รายละเอียดในการคำนวณด้วยวิธีของ Crout Decomposition อ่านได้ในบทที่ 2 หัวข้อที่ 2.4.6) เมื่อ P0, P1, P2, P3 แทนชื่อโหนดบนระบบคลัสเตอร์

กำหนดให้ $A = \begin{bmatrix} 1 & 2 & 1 & 2 \\ 2 & 1 & 2 & 1 \\ 4 & 5 & 1 & 0 \\ 3 & 4 & 0 & 4 \end{bmatrix}_{4x4}$

ขั้นตอนที่ 1 ทำการกระจายข้อมูลของเมตริก A ในแนว Column ไปยังโหนด P0, P1, P2, P3 ซึ่งจะได้ผลดังนี้

P0	P1	P2	P3
1	2	1	2
2	1	2	1
4	5	1	0
3	4	0	4

ขั้นตอนที่ 2 หาสมาชิกของเมตริก L ในหลักที่ 1 จาก $l_{i1} = a_{i1}$ For i=1, 2, 3, …, n

P0	P1	P2	P3
1	2	1	2
2	1	2	1
4	5	1	0
3	4	0	4

ในขั้นตอนนี้จะทำการหา $l_{11}, l_{21}, l_{31}, l_{41}$ จาก $l_{i1} = a_{i1}$ For i=1, 2, 3, …, n เมื่อ n คือขนาดของเมตริกซ์ ดังนั้น $l_{11} = a_{11}$, $l_{21} = a_{21}$, $l_{31} = a_{31}$ และ $l_{41} = a_{41}$ โดยที่ $l_{11}, l_{21}, l_{31}, l_{41}$ เป็นข้อมูลที่อยู่ในโหนด P0 ทำให้ในขั้นตอนนี้ไม่มี Communication เกิดขึ้น ในขณะที่ในขั้นตอนนี้ก็ไม่มีการคำนวณใดๆเกิดขึ้นจึงทำให้ไม่มี Computation เกิดขึ้นเช่นกัน

จำนวน Computation **= 0**

จำนวน Communication **= 0**

ขั้นตอนที่ 3 หาสมาชิกของเมตริก U ในแถวที่ 1 จาก $u_{1j} = \frac{a_{1j}}{l_{11}}$ For j=2, 3, …, n

P0	P1	P2	P3
1	2	1	2
2	1	2	1
4	5	1	0
3	4	0	4

ในขั้นตอนนี้จะทำการหา u_{12}, u_{13}, u_{14} จาก $u_{1j} = \frac{a_{1j}}{l_{11}}$ For j=2, 3, …, n เมื่อ n คือขนาดของเมตริกซ์ ดังนั้น $u_{12} = \frac{a_{12}}{l_{11}}$, $u_{13} = \frac{a_{13}}{l_{11}}$ และ $u_{14} = \frac{a_{14}}{l_{11}}$ โดยที่ u_{12} เป็นข้อมูลที่อยู่ในโหนด P1, u_{13} เป็นข้อมูลที่อยู่ในโหนด P2 และ u_{14} เป็นข้อมูลที่อยู่ในโหนด P3 ซึ่งในการที่จะคำนวณหา u_{12}, u_{13}, u_{14} ได้ต้องใช้ค่า l_{11} ร่วมด้วย แต่ l_{11} เป็นข้อมูลที่อยู่ในโหนด P0 ดังนั้นโหนด P0 จะต้องส่งค่า l_{11} ไปให้โหนด P1, P2 และ P3 ทำให้มี Communication เกิดขึ้น 3 Communications และมี Computation เกิดขึ้น 1 Computation โดย Computationในขั้นตอนนี้เกิดจาก Operator หารที่เกิดขึ้น 1 Operator ในโหนด P1, P2 และ P3 ซึ่งในการนับ Operation ของการประมวลผลแบบขนานจะนับ Operation แต่ละตัวเป็น 1 Operator เท่านั้นหาก Operation นั้น ๆ ได้เกิดขึ้นพร้อม ๆ กันในแต่ละโหนด

จำนวน Computation **= 1**

จำนวน Communication **= 3**

ขั้นตอนที่ 4 หาสมาชิกของเมตริก L ในหลักที่ 2 จาก $l_{ij} = a_{ij} - \sum_{k=1}^{j-1} l_{ik}u_{kj}$ สำหรับ i = j, j+1, ..., n เมื่อ j = 2

P0	P1	P2	P3
1	2	1	2
2	1	2	1
4	5	1	0
3	4	0	4

ในขั้นตอนที่ 2 ได้ทำการหาสมาชิกของเมตริกซ์ L ในหลักที่ 1 ส่วนในขั้นตอนที่ 4 นี้จะทำการหาสมาชิกของ L ในหลักที่ 2 จาก $l_{ij} = a_{ij} - \sum_{k=1}^{j-1} l_{ik}u_{kj}$ สำหรับ i = j, j+1, ..., n เมื่อ j = 2 และ n คือขนาดของเมตริกซ์

ดังนั้น $l_{22} = a_{22} - \sum_{k=1}^{2-1} l_{21}u_{12}$, $l_{32} = a_{32} - \sum_{k=1}^{2-1} l_{31}u_{12}$

และ $l_{42} = a_{422} - \sum_{k=1}^{2-1} l_{41}u_{12}$ โดยที่ l_{22}, l_{32}, l_{42} เป็นข้อมูลที่อยู่ในโหนด P1 เท่านั้น แต่ในการคำนวณหา l_{22} จะต้องใช้ l_{21}, l_{32} จะต้องใช้ l_{31} และ l_{42} จะต้องใช้ l_{41} ช่วยในคำนวณซึ่ง l_{21}, l_{31}, l_{41} เป็นข้อมูลที่อยู่ในโหนด P0 ดังนั้นจึงมี Communication เกิดขึ้น 3 Communication เพราะโหนด P0 ต้องส่ง l_{21}, l_{31}, l_{41} ให้โหนด P1 ซึ่งมี 3 ข้อมูลที่ต้องส่งทำให้เกิด Communication เท่ากับ 3 Communication ขณะที่ Computation ที่ใช้ในการคำนวณหา l_{22} เกิดขึ้น 2 Computation เพราะมี Operator ลบ และคูณ อย่างละ 1 ตัว (เครื่องหมายลบบนหัว Sigma ไม่นับ) และ l_{32} และ l_{42} ก็มีจำนวน Computation เกิดขึ้น 2 เหมือนกันกับการคำนวณหา l_{22} เพราะในขั้นตอนนี้ใช้สมการเดียวกันในการหาสมาชิกของ L ดังนั้น Computation ที่เกิดขึ้นทั้งหมดในขั้นตอนนี้จะมีทั้งหมด 6 Computation เนื่องจากการประมวลผลในขั้นตอนนี้ไม่ได้มีโหนดอื่นช่วยในการประมวลเพราะข้อมูลที่ต้องการหาได้อยู่ในโหนด P1 เพียงโหนดเดียว

จำนวน Computation **= 6**

จำนวน Communication **= 3**

ขั้นตอนที่ 5 หาสมาชิกของเมตริก U จาก $u_{jk} = \dfrac{a_{jk} - \sum_{i=1}^{j-1} l_{ji} u_{ik}}{l_{jj}}$ สำหรับ k = j+1, j+2, …, n เมื่อ j มีค่าเท่ากับ 2

P0	P1	P2	P3
1	2	1	2
2	-3	2	1
4	-3	1	0
3	-2	0	4

ในขั้นตอนที่ 3 ได้ทำการหาสมาชิกของเมตริกซ์ U ในแถวที่ 1 ส่วนในขั้นตอนที่ 5 นี้จะทำการหาสมาชิกของ U ในแถวที่ 2 จาก

$u_{jk} = \dfrac{a_{jk} - \sum_{i=1}^{j-1} l_{ji} u_{ik}}{l_{jj}}$ สำหรับ k = j+1, j+2, …, n เมื่อ j มีค่าเท่ากับ 2 และ n คือขนาดของเมตริกซ์

ดังนั้น $u_{23} = \dfrac{a_{23} - \sum_{i=1}^{2-1} l_{21} u_{13}}{l_{22}}$ และ $u_{24} = \dfrac{a_{24} - \sum_{i=1}^{2-1} l_{21} u_{14}}{l_{22}}$ โดยที่ u_{23} เป็นข้อมูลที่อยู่ในโหนด P2 และ u_{23} เป็นข้อมูลที่อยู่ในโหนด P3 แต่ในการคำนวณหา u_{23} และ u_{24} จะต้องใช้ l_{21}, l_{22} ช่วยในคำนวณ โดย l_{21} เป็นข้อมูลที่อยู่ในโหนด P0 ส่วน l_{22} เป็นข้อมูลในโหนด P1 ดังนั้นจึงมี Communication เกิดขึ้น 4 Communication เพราะโหนด P0 ต้องส่ง l_{21} ให้ P2 และ l_{31} ให้P3 ทำให้เกิด Communication ขึ้น 2 Communication ขณะที่ โหนด P1 ต้องส่ง l_{22} ให้ P2 และP3 ตามลำดับทำให้ทำให้เกิด Communication ขึ้นอีก 2 Communication รวม Communication ที่เกิดในขั้นตอนที่ 4 นี้เท่ากับ 4 Communication และComputation ที่ใช้ในการคำนวณหา u_{23} เกิดขึ้น 3 Computation เพราะมี Operator ลบ, คูณ และหาร อย่างละ 1 ตัว(เครื่องหมายลบบนหัว Sigma ไม่นับ) และในการคำนวณหา u_{24} ก็มีจำนวน Computation เกิดขึ้น 3 Computation แต่เนื่องจากเป็นการประมวลที่เกิดขึ้นพร้อม ๆ Computation ในขั้นตอนนี้จึงเท่ากับ 3

จำนวน Computation **= 3**

จำนวน Communication **= 4**

ขั้นตอนที่ 6 หาสมาชิกของเมตริก L จาก $l_{ij} = a_{ij} - \sum_{k=1}^{j-1} l_{ik} u_{kj}$ สำหรับ i = j, j+1, …, n เมื่อ j มีค่าเท่ากับ 3

P0	P1	P2	P3
1	2	1	2
2	-3	0	1
4	-3	1	0
3	-2	0	4

ในขั้นตอนที่ 2 และ 4 ได้ทำการหาสมาชิกของเมตริกซ์ L ในหลักที่ 1 และ 2 แล้ว ซึ่งในขั้นตอนที่ 6 นี้จะทำการหาสมาชิกของ L ในหลักที่ 3 จาก $l_{ij} = a_{ij} - \sum_{k=1}^{j-1} l_{ik} u_{kj}$ สำหรับ i = j, j+1, …, n เมื่อ j = 3 และ n คือขนาดของเมตริกซ์

ดังนั้น $l_{33} = a_{33} - \sum_{k=1}^{3-1} l_{31} u_{13} - l_{32} u_{23}$

และ $l_{43} = a_{43} - \sum_{k=1}^{3-1} l_{41} u_{13} - l_{42} u_{23}$ โดยที่ l_{33}, l_{43} เป็นข้อมูลที่อยู่ในโหนด P2 เท่านั้น แต่ในการคำนวณหา l_{33} จะต้องใช้ l_{31}, l_{32}, และในการหา l_{43} จะต้องใช้ l_{41} ช่วยในคำนวณ ซึ่ง l_{31}, l_{41} เป็นข้อมูลที่อยู่ในโหนด P0 ดังนั้นจึงมี Communication เกิดขึ้น 2 Communication เพราะโหนด P0 ต้องส่งค่า l_{31}, l_{41} ให้โหนด P2 ซึ่งมี 2 ข้อมูลที่ต้องส่งทำให้เกิด Communication เท่ากับ 2 Communication และโหนด P1 ต้องส่ง l_{32}, l_{42} ให้โหนด P2 เช่นกันจึงทำให้เกิด Communication ขึ้นอีก 2 Communication ทำให้ Communication ในขั้นตอนที่ 6 นี้มีทั้งหมด 4 Communication และ Computation ที่ใช้ในการคำนวณหา l_{33} เกิดขึ้น 4 Computation เพราะมี Operator ลบ และคูณ อย่างละ 2 ตัว(เครื่องหมายลบบนหัว Sigma ไม่นับ) และ l_{42} ก็มีจำนวน Computation เกิดขึ้น 4 เหมือนกันกับการคำนวณหา l_{33} เพราะในขั้นตอนนี้ใช้สมการเดียวกันในการหาสมาชิกของ L ดังนั้น Computation ที่เกิดขึ้นทั้งหมดในขั้นตอนนี้จะมีทั้งหมด 8 Computation เนื่องจากการประมวลผลในขั้นตอนนี้ไม่ได้มีโหนดอื่นช่วยในการประมวลเพราะข้อมูลที่ต้องการหาได้อยู่ในโหนด P2 เพียงโหนดเดียว

จำนวน Computation **= 8**

จำนวน Communication **= 4**

ขั้นตอนที่ 7 หาสมาชิกของเมตริก U จาก $u_{jk} = \frac{a_{jk} - \sum_{i=1}^{j-1} l_{ji} u_{ik}}{l_{jj}}$ สำหรับ k = j+1, j+2, …, n เมื่อ j มีค่าเท่ากับ 3

P0	P1	P2	P3
1	2	1	2
2	-3	0	1
4	-3	-3	0
3	-2	-3	4

ในขั้นตอนที่ 3 และ 5 ได้ทำการหาสมาชิกของเมตริกซ์ U ในแถวที่ 1 และ 2 แล้ว ซึ่งในขั้นตอนที่ 7 นี้จะทำการหาสมาชิกของ U ในแถวที่ 3 จาก $u_{jk} = \frac{a_{jk} - \sum_{i=1}^{j-1} l_{ji} u_{ik}}{l_{jj}}$ สำหรับ k = j+1, j+2, …, n เมื่อ j มีค่าเท่ากับ 3 และ n คือขนาดของเมตริกซ์

ดังนั้น $u_{34} = \frac{a_{34} - \sum_{i=1}^{3-1} l_{31} u_{14} - l_{32} u_{24}}{l_{33}}$ โดยที่ u_{34} เป็นข้อมูลที่อยู่ในโหนด P3 แต่ในการคำนวณหา u_{34} จะต้องใช้ l_{31}, l_{32}, l_{33} ช่วยในคำนวณ โดย l_{31} เป็นข้อมูลที่อยู่ในโหนด P0 ส่วน l_{32} เป็นข้อมูลในโหนด P1 และ l_{33} เป็นข้อมูลในโหนด P2 ดังนั้นจึงมี Communication เกิดขึ้น 3 Communication เพราะโหนด P0 ต้องส่ง l_{31} ให้ P3, l_{32} ให้ P3 และ l_{33} ให้ P3 ทำให้เกิด Communication ขึ้น 3 Communication ดังนั้นในขั้นตอนที่ 7 นี้จึงมี Communication ทั้งหมด 3 Communication และComputation ที่ใช้ในการคำนวณหา u_{34} เกิดขึ้นทั้งหมด 5 Computation เพราะมี Operator ลบ, คูณ อย่างละ 2 ตัวและหาร อีก 1 ตัว (เครื่องหมายลบบนหัว Sigma ไม่นับ)

จำนวน Computation **= 5**

จำนวน Communication **= 3**

ขั้นตอนที่ 8 หาสมาชิกของเมตริก L จาก $l_{ij} = a_{ij} - \sum_{k=1}^{j-1} l_{ik} u_{kj}$ สำหรับ i = j, j+1, …, n เมื่อ j มีค่าเท่ากับ 4

P0	P1	P2	P3
1	2	1	2
2	-3	0	1
4	-3	-3	5/3
3	-2	-3	4

ในขั้นตอนที่ 2, 4 และ 6 ได้ทำการหาสมาชิกของเมตริกซ์ L ในหลักที่ 1, 2 และ 3 แล้ว ซึ่งในขั้นตอนที่ 8 นี้จะทำการหาสมาชิกของ L ในหลักที่ 4 จาก $l_{ij} = a_{ij} - \sum_{k=1}^{j-1} l_{ik} u_{kj}$ สำหรับ i = j, j+1, …, n เมื่อ j = 4 และ n คือขนาดของเมตริกซ์

ดังนั้น $l_{44} = a_{44} - \sum_{k=1}^{4-1} l_{41} u_{14} - l_{42} u_{24} - l_{43} u_{34}$

และ $l_{44} = a_{44} - \sum_{k=1}^{4-1} l_{41} u_{14} - l_{42} u_{24} - l_{43} u_{34}$ โดยที่ l_{44} เป็นข้อมูลที่อยู่ในโหนด P3 เท่านั้น แต่ในการคำนวณหา l_{44} จะต้องใช้ l_{41}, l_{42}, l_{43} ซึ่ง l_{41} เป็นข้อมูลที่อยู่ในโหนด P0, l_{42} เป็นข้อมูลที่อยู่ในโหนด P1 และ l_{43} เป็นข้อมูลที่อยู่ในโหนด P2 ดังนั้นจึงมี Communication เกิดขึ้นทั้งหมด 3 Communication เพราะโหนด P0 ต้องส่งค่า l_{41} ให้โหนด P3, โหนด P1 ต้องส่งค่า l_{42} ให้โหนด P3 และโหนด P2 ต้องส่งค่า l_{43} ให้โหนด P3 ทำให้ Communication ในขั้นตอนที่ 8 นี้มีทั้งหมด 3 Communication และ Computation ที่ใช้ในการคำนวณหา l_{44} เกิดขึ้นทั้งหมด 6 Computation เพราะมี Operator ลบ และคูณ อย่างละ 3 ตัว(เครื่องหมายลบบนหัว Sigma ไม่นับ) ดังนั้น Computation ที่เกิดขึ้นทั้งหมดในขั้นตอนนี้จะมีทั้งหมด 6 Computation

จำนวน Computation **= 6**

จำนวน Communication **= 3**

หลังจากเสร็จสิ้นขั้นตอนทั้ง 8 จะทำให้ได้ Computation และ Communication ดังนี้

P0	P1	P2	P3
1	2	1	2
2	-3	0	1
4	-3	-3	5/3
3	-2	-3	5

หลังจากเสร็จสิ้นทั้ง 8 ขั้นตอนแล้วจะทำการหาผลรวมของ Computation และ Communication โดยทำการรวม Computation และ Communication ที่เกิดขึ้นจากขั้นตอนทั้ง 8 เข้าด้วยกัน

รวมจำนวน Computation = **29**

รวมจำนวน Communication = **20**

ภาคผนวก ข
คำสั่งพื้นฐานของ MPI

ภาคผนวก ข
คำสั่งพื้นฐานของ MPI

ในส่วนนี้จะเป็นการอธิบายรูปแบบคำสั่งพื้นฐานของ MPI ที่ใช้สำหรับพัฒนาโปรแกรมเพื่อให้ทำงานแบบขนานบนระบบคลัสเตอร์

1. **int MPI_Init(int *argc, char ***argv)** เป็นคำสั่งที่ใช้ประกาศเพื่อเริ่มต้นการทำงาน

2. **int MPI_Comm_size (MPI_Comm comm, int *size)** เป็นคำสั่งที่ใช้ประกาศว่าจะมีเครื่องจำนวนกี่เครื่องที่จะทำงานร่วมกัน

3. **int MPI_Comm_rank (MPI_Comm comm, int *rank)** เป็นคำสั่งที่แต่ละเครื่องจะเรียกใช้เพื่อให้ทราบว่าถึงตำแหน่งของตัวมันเอง (rank) สำหรับการประมวลผล

4. **int MPI_Send(void *buf, int count, MPI_Datatype datatype, int dest, int tag, MPI_Comm comm)** เป็นสำสั่งที่ใช้สำหรับส่งข้อมูลไปยังเครื่องตามที่ได้ระบุปลายทางไว้

   buf = แอดเดรสของ ข้อมูลหรือบัฟเฟอร์

   count = จำนวนข้อมูลที่จะส่ง

   datatype = ชนิดของข้อมูล เช่น MPI_FLOAT สำหรับตัวเลข floating point , MPI_INT สำหรับตัวเลขจำนวนเต็ม

   dest = rank ของโปรเซสที่จะส่งข้อมูลให้

   tag = ตัวเลขที่ผู้ใช้กำหนด อาจใช้ช่วยเช่น บอก ชนิดข้อมูล ก็ได้

   communicator = ปกติจะเป็น MPI_COMM_WORLD ซึ่งเป็น default communicator

5. **int MPI_Recv (void *buf, int count, MPI_Datatype datatype, int source, int tag, MPI_Comm comm, MPI_Status *status)** เป็นคำสั่งที่ใช้สำหรับรับข้อมูลจากเครื่องที่ได้ทำการส่งข้อมูลด้วยคำสั่ง MPI_Send

   buf = แอดเดรสของ ข้อมูลหรือบัฟเฟอร์

   count = จำนวนข้อมูลที่รับได้

   datatype = ชนิดของข้อมูล เช่น MPI_FLOAT สำหรับตัวเลข floating point , MPI_INT สำหรับตัวเลขจำนวนเต็ม

   source = rank ของโปรเซสที่จะรับข้อมูล

tag = ตัวเลขที่ผู้ใช้กำหนด อาจใช้ช่วยเช่น บอก ชนิดข้อมูล ก็ได้

communicator = ปกติจะเป็น MPI_COMM_WORLD ซึ่งเป็น default communicator status = สถานะของการรับข้อมูล

6. **MPI_Finalize()** เป็นคำสั่งที่ใช้สำหรับประกาศว่าจบการทำงาน (ใช้คู่กับ MPI_Init)

7. **mpicc** เป็นคำสั่งที่ใช้สำหรับคอมไพล์โปรแกรมที่เขียนจาก MPI Library สำหรับภาษา C

8. **mpirun** เป็นคำสั่งที่ใช้รันโปรแกรมที่ได้ทำการคอมไฟล์แล้วเท่านั้น

หมายเหตุ หาต้องการทราบถึงรูปแบบคำสั่งอื่นของ MPI สามารถดูได้จากเว็บไซต์ http://www-unix.mcs.anl.gov/mpi/mpich/

ตัวอย่าง โปรแกรมแบบขนานด้วย MPI ซึ่งในตัวอย่างโค้ดข้างล่างนี้จะเป็นโปรแกรม Hello Word แบบขนาน

```
/* ---------------------------------------------------------------------- */
/* --        Example for parallel 'Hello world!' program.     -- */
/*----------------------------------------------------------------------*/
#include <stdio.h>
#include "mpi.h"

int main (int argc, char *argv[])
{
  int size, rank;

  MPI_Init (&argc, &argv);
  MPI_Comm_size (MPI_COMM_WORLD, &size);
  MPI_Comm_rank (MPI_COMM_WORLD, &rank);
  printf ("Hello world! from processor (%d/%d)\n", rank+1, size);
  MPI_Finalize();
return 0;
}
```

เมื่อนำโปรแกรมข้างต้นนี้ไปทำการคอมไพล์และทำการรันจะได้ผลดังนี้

Hello world! from processor (2/4)

Hello world! from processor (1/4)

Hello world! from processor (3/4)

Hello world! from processor (4/4)

ประวัติผู้วิจัย

ชื่อ	นายอนิรุธ สืบสิงห์
ประวัติการศึกษา	ระดับมัธยมศึกษา
	โรงเรียนศรีปทุมพิทยาคาร
	อ.เมือง จ.อุบลราชธานี
	ระดับปริญญาตรีหลักสูตรวิทยาศาสตรบัณฑิต
	สาขาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์
	สถาบันราชภัฏอุบลราชธานี ปี พ.ศ. 2543
ประวัติการวิจัย	ทุนสนับสนุนการทำวิทยานิพนธ์
	ทุนพัฒนาอาจารย์สาขาขาดแคลน สกอ.
	คณะบริหารศาสตร์ มหาวิทยาลัยอุบลราชธานี ปี 2547
ประวัติการทำงาน	พ.ศ. 2544-2546
	ผู้ดูแลระบบเครือข่าย บริษัทราชบุตรเทเลคอม
	อ.เมือง จ.อุบลราชธานี
	พ.ศ. 2546-2547
	นักวิชาการคอมพิวเตอร์ สำนักคอมพิวเตอร์และเครือข่าย
	มหาวิทยาลัยอุบลราชธานี
สามารถติดต่อได้ที่	
โทรศัพท์	+6691405590
E-mail	anirut.s@gmail.com